Luca Stefano Accomazzi

Il Manuale dell' Apple II GS

HOME E PERSONAL COMPUTER

GRUPPO EDITORIALE
JACKSON

Il Manuale dell'Apple II GS

Luca Stefano Accomazzi

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

INDICE

PREFAZIONE

1. Salve a tutti

Grazie di aver scelto questo libro per la vostra biblioteca: farò del mio meglio perché non ve ne pentiate.
Permettetemi di presentarmi, dato che forse non mi conoscerete. Mi chiamo Luca Accomazzi, e sono conosciuto anche come Mister Akko.

In questo libro vi guiderò alla scoperta dell'ultimo nato di casa Apple, lo stupefacente Apple IIgs, la punta di diamante della produzione di Casa Apple.
La maggior parte del libro è dedicata a spiegare come è fatto, come funziona, come usare al meglio il computer. D'altra parte due capitoli sono interamente dedicati al problema della programmazione, ai trucchi più potenti, ai metodi più flessibili che rendono la vita del programmatore più semplice.

Non posso garantirvi che dirò tutto quello che c'è da dire. Come sempre nel caso di macchine Apple create dallo stupefacente Steve Wozniak, le potenzialità dello *hardware* (la macchina fisica) sono tali e tante che nuove possibilità vengono scoperte ogni giorno.
Non posso neppure garantirvi che tutto quello che troverete in questo libro vi interesserà direttamente. La massa di informazioni che ho accumulato in queste pagine è davvero notevole ma è anche eterogenea, dato che tocca tutti gli aspetti della macchina: se leggerete il libro dalla prima all'ultima pagina trovandolo indistintamente interessante vi sarete guadagnati il titolo di *hacker*, appassionati ed inguaribili: vostra moglie diverrà presto una vedova bianca, venendo tradita a favore di una bionda platino fatta di silicio, e finirete per acquisire una abbronzatura dovuta alle irradiazioni del monitor.

Una cosa posso promettervela, e la prometto sin da questo punto. In questo libro troverete una massa di informazioni che vi erano sconosciute (a meno che vi chiamiate Steve Wozniak). E, anche se vostra moglie **non** dimenticherà l'aspetto della vostra faccia, avrete comunque imparato a conoscere le potenzialità

del computer che – spero – avete scelto saggiamente di comprare.

2. Credo di essermi innamorato di nuovo

La mia prosa appare normalmente sulle pagine di **Bit**, una rivista del Gruppo Editoriale Jackson. Per tutta la sua ahimé troppo breve vita, inoltre, i miei scritti hanno regolarmente infestato **SuperApple**, una rivista dedicata interamente alle macchine di casa Apple, e proprio sul primo numero di SuperApple appariva un mio articolo che ancora oggi molti dei miei venticinque lettori amano citare quando mi incontrano. L'articolo si chiamava "Innamorarsi di un computer", e spiegava perché il sottoscritto ha sviluppato quella morbosa passione che ho appena ricordato.

In quell'articolo dicevo che...

Innanzitutto, Apple IIc (familiarmente "Ciccio") ha più programmi pronti di qualunque altro computer, e questo dovrebbe essere il fattore più considerevole quando scegliete un computer; comprereste un televisore che consuma pochissimo, può memorizzare un milione di canali, tridimensionale, se questo non potesse ricevere nessun programma televisivo? No, perché sarebbe una spesa inutile.

Esiste in commercio (mi meraviglio di come non sia ancora fallita la ditta) un PC che va gridando di avere "centinaia di programmi pronti". E allora? Ciccio ne ha centomila, dei quali venticinquemila commercialmente prodotti, distribuiti e supportati, secondo le stime.

In secondo luogo, dovete tener conto che non si può proteggere un programma scritto per Apple II. C'è una simpatica cosuccia chiamata **Monitor** *a bordo di Ciccio, con la quale è possibile entrare in qualunque programma commerciale, studiarlo e modificarlo; questo significa che se qualcuno "inventa" qualcosa di simpatico, diciamo un nuovo modo di animare disegni tridimensionali, o di produrre musica, e lo usa in un programma, chiunque può andare a vedere come il programmatore ha fatto, ed usare lo stesso algoritmo nei propri programmi.*

Detto fra di noi, aprire, sproteggere, esaminare e modificare a proprio capriccio un programma commerciale è anche il passa-

tempo più divertente, gratificante, istruttivo che possiate intraprendere; quelli che si dilettano così vanno sotto il nome di "hackers", e non vanno confusi con il loro alter-ego maligno, i pirati del software, che copiano i programmi altrui e li rivendono (dunque, sono dei ladri, che rubano il lavoro altrui).

Nell'articolo proseguivo ammettendo a denti stretti le tre limitazioni di Apple IIc rispetto ai concorrenti: processore a soli 8 bit, grafica e suono non eccelsi, poca memoria RAM.

Bene, eccoci qui a regolare i conti: Apple IIgs ha una grafica incredibile, da contare tra le prime due disponibili sul mercato. Il suo suono è il migliore in assoluto, anni luce avanti al secondo in classifica. Il processore può usare più memoria di ogni altro (ex aequo con il 68000 di Macintosh) ed è un sedici bit; è probabilmente il più flessibile e il più facile da programmare di tutti.

E allora, buttate via indignati gli altri computer che vi capita di vedere, ridete a crepapelle dei loro malcapitati possessori, fate largo ad Apple IIgs, fate largo alla fenice!

3. Fate largo alla fenice!

La fenice? Sì, la fenice. Phoenix è uno degli innumerevoli nomi che il progetto Apple IIgs ha avuto prima di venire battezzato definitivamente.

Il primo nome è stato Apple IIx, dove x stava per "extra". Poi i cervelloni della divisione marketing della Apple hanno fuso le proprie meningi alla caccia di un migliore nome di battaglia, passando per Cortland (una qualità di mela californiana), Vegas (la sigla, credo, di Very Enhanced Graphics And Sound), Brooklin (perché è il ponte verso un nuovo mondo di personal computer) e persino Rambo — trema, Big Blue!

Tra tutti i nomi, il mio preferito è senz'altro Phoenix: perché con l'Apple IIgs la serie Apple II già data più volte per morta rinasce dalle proprie ceneri, pronta a restare protagonista della scena per altri dieci anni e più.

Così, spesso me lo sentirete chiamare Phoenix: ma è sempre lui, il nostro amico Apple II Graphics & Sound.

4. Avvertenze

Il nocciolo del libro è una spiegazione di come può funzionare l'entità logica Apple IIgs (non l'entità fisica, perché questo non è un libro per pochi ingegneri elettronici, non l'entità informatica perché questo libro deve servire anche ai non programmatori) e come sono state realizzate le sue dotazioni. La mia opinione è che la conoscenza di come funziona il computer, a prescindere dai dettagli tecnici, fà la differenza tra un utente inesperto che si dispera quando qualcosa non va come dovrebbe e un utente esperto che sa riparare i piccoli guasti, adattare il sistema alle proprie esigenze (e non viceversa), usare al pieno delle sue – spettacolari – capacità il proprio investimento di soldi e tempo. E, prima ancora di questo, soddisfa quella curiosità di sapere che ci viene come eredità della razza umana.

Il mio impegno di spiegare quanto più possibile vuol dire che qualche punto isolato di questo libro probabilmente vi risulterà oscuro, e qualche altro pur comprendendolo vi sembrerà irrilevante. Non poteva essere altrimenti, dato che intendo fornire notizie utili ai principianti come agli esperti: probabilmente ad una seconda lettura qualche altro tassello entrerà al suo posto nello schema complessivo.

L'ordine in cui inserire i capitoli mi ha dato qualche grattacapo. Mi sono trovato nei panni dell'istruttore che deve spiegare come funziona una automobile: inizierò a spiegare come si usa, ma come posso spiegare i trucchi che si possono fare col cambio se non ho ancora toccato l'argomento degli ingranaggi? La matassa non ha un bandolo, perché tutti gli argomenti possono venire compresi appieno solo conoscendo già tutti gli altri.

Ne è uscito un libro che non va letto una sola volta, perché si perderebbero tutti i riferimenti in avanti, né tanto meno un manuale da consultare sui soli argomenti di interesse, ma piuttosto un ibrido tra i due.

Se non siete degli esperti, delle vecchie volpi che hanno già usato per anni i vecchi modelli di Apple II, leggete questo libro per la prima volta dall'inizio alla fine, e passate poi a rileggerne i singoli capitoli man mano che ne sentite il bisogno: ogni volta scoprirete qualche cosa di nuovo, qualche nuovo pezzo del puzzle entrerà al suo giusto posto.

Attenzione! Per evitare le noiose ripetizioni, ho evitato di ri-

spiegare i termini di base che si incontrano usando Apple IIgs: parole come "disco", "finestra", "applicazioni". Se ne sapete pochino, se Apple IIgs è il vostro primo computer, se non avete preso confidenza con Phoenix, la cosa migliore che possiate fare è leggere per primo il Manuale Utente che avete trovato insieme al vostro computer. Poi riprendete in mano questo libro.

5. Ringraziamenti

Cosa sarebbe una prefazione senza i ringraziamenti e le dediche?

Nel dare alle stampe questo libro voglio ringraziare mia madre, che non ha mai mancato di illuminarmi sulle conseguenze nefaste del dedicare più tempo a questo progetto che ai miei esami universitari. E a nonna Clementina Rebecca, che mi esorta invece a passare più tempo ad inseguire gonnelle (diavolo di una nonna!) Devo poi ricordare Nicola Lepetit e Andrea Rossi Cenotti, che ho convinti ad acquistare un Apple IIc pochissimi mesi prima che Apple IIgs apparisse sul mercato, e che ciononostante non mi hanno ucciso.

Ringrazio Carlo Bocchetti, che tanto per cambiare **non** ha scritto questo libro con il sottoscritto. Tre a quattro, Bock!

Ringrazio Carlo Magnaghi, il più grande mago dello hardware Apple in Italia, per le illuminanti chiacchierate sul funzionamento della RAM. Ringrazio Enrico Colombini, e ringrazio Emilio Re Garbagnati per avermi illuminato sui segreti della comunicazione degli Apple con le stampanti.

Più di tutti ringrazio Lui, il mio Apple II, che mi ha dato il permesso di spiegarvi i segreti della sua stupefacente esistenza, dopo avermeli lasciati scoprire uno ad uno in numerose notti d'insonnia alla tastiera. Faremo ancora lunghi viaggi insieme, mela mia...

DENTRO LA SCATOLA

Cosa c'è dentro quella scatola grigio platino, sia da un punto di vista fisico che logico? È quanto scopriamo nel primo capitolo, concludendo poi con i dati tecnici dell'Apple IIgs commentati.

1.1 Visita guidata dentro IIgs

Suppongo che in questo momento voi abbiate davanti a voi il vostro Apple IIgs. Bene, apritelo: è un lavoretto da venti secondi e non richiede nessuno strumento.
Se il computer è acceso, spegnetelo e aspettate trenta secondi prima di staccare la spina dalla presa di corrente.
Staccate il cordone dell'alimentazione elettrica dall'alloggiamento sul retro del computer. Premete contemporaneamente le due linguette in alto a destra e sinistra sul retro del computer e sollevate; durante i primi tentativi forse sollevando il coperchio alzerete anche il resto del computer. In quel caso fatevi aiutare da qualcuno che spinga il corpo verso il basso mentre voi alzate il coperchio.

Quella scatola argentata e forata che vedete sulla sinistra è l'alimentatore trasformatore del computer. Toccatelo: in questo modo vi libererete dell'eventuale elettricità statica del vostro corpo ed eviterete di danneggiare i componenti interni. È possibile spostare anche quello, se volete, tirando il blocco di plastica

che lo fissa sulla parte anteriore del computer sino a liberarlo, e poi spostando in alto e avanti.

Guardiamo dal davanti verso il fondo: parte dei componenti sulla piastra verde (che si chiama **piastra madre** o **motherboard**) sono nascosti dall'alimentatore, ma i più importanti sono in vista.

Davanti a sinistra una serie di componenti neri rettangolari sono identificati dalla scritta **Standard RAM:** sono i 128 Kbyte di memoria ad accesso casuale usati con i programmi creati per i modelli precedenti della serie II. Spostati al centro troviamo altri **chip** identificati come **Sound RAM:** un banco di 64 Kbyte dedicati al processore sonoro, il componente che si occupa di sintetizzare musica, suono e voce. Il **processore sonoro Ensoniq** è il chip più grosso sulla destra, e permette di usare trentadue oscillatori per quindici voci stereofoniche. Il suono viene diretto all'altoparlantino che si trova all'estrema sinistra (nascosto sotto l'alimentatore) oppure a una cuffia stereofonica che venga collegata sul retro del computer.

Sulla destra poco più in là del processore sonoro c'è una fessura circondata da contatti metallici ed affiancato dalla scritta **Memory Expansion**. Può darsi che questa fessura, normalmente chiamata **slot**, contenga una scheda verticale con altri componenti.

Questo slot viene usato per aumentare la memoria del calcolatore: ne riparleremo nel prossimo paragrafo.

Di fianco allo slot vediamo due chip coperti da una etichetta bianca, uno quadrato ed uno più grande e rettangolare, che contiene la **ROM** del computer: 128 Kbyte di memoria permanente comprendente il System Monitor, il ToolBox, i Manager, diversi Device Driver, la libreria matematica SANE, il linguaggio basic Applesoft. In breve, tutte quelle entità di cui parleremo nel resto del libro.

Un chip nero un po' più lungo della ROM le sta a fianco: è il punto nevralgico del sistema, il cervello del computer, la sua **CPU, Central Processing Unit** o unità di elaborazione centrale.

Il chip, marcato dalla scritta G65SC816 ed altre lettere e cifre, è il **microprocessore** dell'Apple IIgs, cioè quel componente che esegue i programmi.

Tutti i microprocessori sono battezzati con un numero. Ad esempio, il cuore del PC IBM è chiamato 8086, quello dell'Apple Macintosh è un 68000, quello del Commodore 64 un 6510, quel-

lo dello Spectrum uno Z80. Il processore di Apple IIgs è chiamato normalmente 65816, tralasciando le altre lettere. Lavora a due velocità differenti a vostra scelta, ad otto oppure sedici bit, ed è servito dagli altri chip e processori che si trovano dentro il computer.

Un chip molto importante, di forma quadrata, è quasi del tutto nascosto dall'alimentatore sulla sinistra un po' più in basso del 65816. È il **Mega II**, il componente che si preoccupa di garantire la compatibilità del nostro elaboratore con tutti i modelli precedenti della serie Apple II.

In quel singolo quadrato nero sono state unite le funzionalità di circa centocinquanta circuiti integrati dell'Apple II+, trenta chip dell'Apple IIe o cinque *custom chip* dell'Apple IIc. L'orgoglio della Apple Computers nel dichiarare che "abbiamo ridotto il nostro più grande computer in questo piccolo chip" è ben giustificata.

Direttamente sopra Mega II stanno altri due chip quadrati affiancati (quello a sinistra è nascosto dall'alimentatore). Il più a destra è lo **FPI, Fast Processor Interface**, il componente che predigerisce i dati per il 65816 e che è responsabile del funzionamento delle capacità aggiunte a Phoenix rispetto ai suoi predecessori. Lavora a maggior velocità rispetto agli altri componenti.

Alla stessa altezza dal lato opposto, sulla destra, sopra la scritta "Apple IIgs", quattro piccoli chip sono chiamati **Fast RAM:** altri 128 Kbyte di RAM a disposizione del computer e più veloci delle precedenti.

Ci resta solo da osservare la fila di componenti che si trovano immediatamente al di sotto della fila di sette slot numerati.

All'estrema sinistra, sotto l'alimentatore, sta un orologio elettronico che resta in funzione anche quando il computer è staccato dalla presa di corrente, grazie alla batteria argentea con la scritta FOX.

Al centro vediamo una presa di plastica bianca con sedici forellini chiamata **Game J21:** serve a collegare le unità per videogiochi, joystick e paddle, che furono creati per i più vecchi Apple II. I modelli più recenti, infatti, possono venire attaccati direttamente sul retro del computer.

Alla sua destra un componente quadrato non molto grande è chiamato **Slot Maker:** si tratta del controllore degli slot, che permette ad ogni tipo di periferica, di dispositivo elettronico o elettrico di essere collegato e controllato dal computer. Sotto la gui-

da di Mega II, Slot Maker permette l'uso di tutte le centinaia di schede differenti create per i precedenti modelli di Apple II.

Alla sinistra del connettore per i giochi si trova un piccolo processore che è probabilmente la più geniale invenzione tra quelle che costituiscono il computer, una perla di ingegneria chiamata **IWM, Integrated Wozniak Machine.** È il gestore delle unità a disco, che permette di usare differenti lettori di dischetti magnetici nello stesso modo veloce e preciso. Deve il suo nome al suo creatore, Steve Wozniak, uno dei due co-creatori del primo Apple II nel 1976; IWM permette di usare più dischi anche contemporaneamente (in teoria sino a 128); i dischi collegati sul retro di Apple IIgs (ne parleremo nel capitolo 7) sono governati in modo identico nonostante le numerose differenze che li distinguono. La gestione del disco realizzata da IWM in unione con il software chiamato SmartPort consente di usare anche uno pseudo disco RAM (cioè parte della memoria del computer viene usata come un disco velocissimo), un disco ROM (memoria permanente che contenga i programmi di uso più frequente), e può governare sino a due terabyte di dati per ogni disco: in cifra estesa, si tratta di 2.199.023.255.552 caratteri, una quantità di spazio nella quale troverebbero spazio più di sette milioni di libri come questo.

L'unione di SmartPort e IWM consente di gestire sino a sedici unità a disco per ogni connettore. Come vedremo, sono le limitazioni fisiche dell'alimentatore che costringono ed a limitare le capacità teoriche del computer a "sole" sei unità a disco in tutto.

1.2 La memoria

Un computer è un elaboratore di dati che segue le istruzioni specificate in un **programma** per eseguire i suoi compiti. Sia il programma che i suoi dati vengono normalmente incisi su un disco magnetico per venire venduti e conservati tra un utilizzo ed il successivo.

Sia il programma che i dati sui quali si vuole che il computer lavori debbono essere portati entro il computer, letti tramite l'unità a disco nella memoria del computer.

Apple IIgs nasce con 256 Kbyte di memoria RAM incorporati, ma questa quantità di memoria può venire facilmente aumentata. Il nostro computer è in grado di gestire sino a 16 Mbyte di

memoria (cioè 16.384 Kbyte), ma il suo sistema operativo (il programma che ha il controllo del corpo fisico della macchina e che permette l'esecuzione di tutti gli altri programmi) attualmente non permette di usare più di 8 Megabyte, la metà di quella cifra.
Nel momento in cui scrivo, inoltre, non sono disponibili schede di espansione di memoria oltre i 4 Megabyte: si tratta comunque di un quantitativo di memoria ritenuto incredibile sino a pochi anni fa e comunque superiore ad ogni altro personal computer di qualsiasi marca che il sottoscritto conosca.

Quanta memoria RAM può davvero servire? Il suo costo, pur non essendo enorme, non è neppure trascurabile, e quindi la scelta di una quantità necessaria e sufficiente di memoria è un argomento di importanza.
Tenendo presente che la differenza nel prezzo tra un Apple IIgs con 256 Kbyte ed un IIgs con il doppio di memoria è di sole centomila lire, vale certamente la pena di rivolgersi al secondo.
Le ulteriori aggiunte sulla scheda di espansione, a blocchi di 256 Kbyte, hanno un prezzo piuttosto elevato: ma in un buon negozio di componenti elettronici, acquistate le RAM 41256 da 256 Kbit (attenzione, chilobit e non chilobyte) con tempo d'accesso di 150 nanosecondi. Hanno un costo di poco più di tremila lire ognuna, e con 24 di queste potete portare un Apple IIgs da 512 Kbyte a 1.280 Kbyte: cioè con una spesa di 75.000 lire.
Per duecentomila lire in più (comprendendo il costo iniziale della scheda Apple) avrete un computer da 1.280 Kbyte di memoria RAM invece che soli 256. Va anche tenuto presente che la memoria non va sprecata, grazie al Memory Manager, e che molti programmi già da ora la possono usare per intero – con in testa Apple Works, il diffusissimo programma Tre per Te, dalla versione 1.4 in poi.

1.3 Le periferiche

Apple IIgs può collegare direttamente a sé una stampante, un modem per comunicare con altri computer via telefono, fino a sei differenti unità a disco in serie, ed usare la rete di comunicazione Apple Talk per collaborare con più stampanti, con altri Apple IIgs, con Apple Macintosh, con i personal computer IBM.
Delle unità a disco ci occuperemo nel capitolo 6; quanto al modem non otterrà dello spazio qui, in quanto se ne avete biso-

gno con ogni probabilità sapete già tutto quanto potrei dire in uno spazio limitato. Ma è molto interessante quello che si può fare con le stampanti, e ancora di più le possibilità messe a disposizione con Apple Talk.

Apple IIgs può usare direttamente ogni tipo di stampante che comunichi con lui in modo **seriale**, cioè un bit alla volta. Molte stampanti hanno come standard la comunicazione **parallela**, cioè un byte alla volta: per collegarle ad Apple IIgs è necessario allora dotare il computer oppure la stampante di una scheda (chiamata **interfaccia**) di comunicazione: potrete inserire una interfaccia parallela nello slot 1 del computer, ed in questo modo le apparecchiature comunicheranno in parallelo: oppure inserite una scheda di interfaccia seriale RS232C nella stampante, per fare comunicare computer e stampante serialmente.

Per far riconoscere dal computer una interfaccia parallela è necessario intervenire dal Pannello di Controllo della macchina: ci ritorneremo nei dettagli nel prossimo capitolo.

Apple Computers produce delle stampanti seriali che sono le più adatte all'Apple IIgs, come possiamo aspettarci. In passato, le stampanti Apple erano scadenti, e si preferiva adottare una meno costosa e più veloce Epson, o Tally, o Star od altre marche ancora. Oggi le stampanti Apple sono all'avanguardia, ed hanno costi piuttosto competitivi.

C'è inoltre una ragione molto importante che consiglia l'acquisto delle stampanti Apple, a prescindere da fattori di costo e potenza: le stampanti devono essere guidate dal computer quando stampano grafici, attraverso un programma opportuno chiamato il **printer driver**. Ovviamente esistono dei printer driver per le stampanti Apple, ma al momento ben poche stampanti non-Apple possono essere usate con i più recenti programmi, perché non sono stati creati i printer driver per loro. A meno che non vi venga messo a disposizione il printer driver per la stampante non Apple che state acquistando, evitate di comprarla: vi ritrovereste con una stampante a mezzo servizio che molti programmi non possono usare.

I printer driver sono dei file che vanno copiati sul disco di lancio del sistema operativo ProDOS 16 nella subdirectory /nomedisco/SYSTEM/DRIVERS in modo che il computer possa usare la stampante con i programmi creati per Apple IIgs. Sul disco di sistema che vi viene fornito insieme al computer sono contenuti i driver per le stampanti Apple Imagewriter ed Apple Laserwriter oltre che per la rete Apple Talk.

Una stampante seriale dedicata esclusivamente a servire il vostro Apple IIgs va collegata sul retro del computer al connettore indicato con il simbolo della stampante. Nell'illustrazione in questa pagina sono indicati i collegamenti da effettuare per collegare Phoenix con una stampante: naturalmente sono in vendita cavetti pressofusi preparati appositamente, ma al momento in cui scrivo non sono disponibili cavi prefabbricati per collegare Apple IIgs con le stampanti con uscita DB25, quella normalmente usata dalle case costruttrici – esclusa Apple che preferisce la presa Minidin8 (la stessa usata nel computer).

Non temete di poter danneggiare il vostro computer costruendo malamente un cavetto: alcuni circuiti di protezione impediscono al computer di accendersi quando le connessioni potrebbero mettere in pericolo la circuiteria interna. È comunque necessario quando si fanno le prove accendere prima la stampante del computer e mai collegare o scollegare i dispositivi a computer acceso.

È anche possibile far usare una stampante da più calcolatori (ovviamente non in contemporanea, ma uno la volta) senza dover attaccare e staccare i fili ogni volta, grazie alla rete Apple Talk.

Rete Apple Talk è il nome con il quale ci riferiamo ad una serie di congegni concreti (hardware) e di programmazione (software) che permettono a più calcolatori di comunicare tra di loro e di condividere l'uso di periferiche come le stampanti. È possibile mettere una sola stampante al servizio contemporaneamente di una decina e più di Apple IIgs e altre macchine inferiori: il che è consigliabile in un ufficio informatizzato, dove anziché comprare una stampante per ogni posto di lavoro diventa possibile scegliere una sola stampante di grande potenza, versatilità, nitidezza... e costo... come la Laser Writer.

Ogni apparecchio collegato alla rete viene detto un **nodo** e deve disporre di un piccolo trasmittente collegato, del costo di meno di centomila lire. I computer che non sono stati creati per usare Apple Talk, come i PC IBM, devono anche venire dotati di una interfaccia apposita.

Con l'Apple IIgs il nodo va collegato sul retro del computer all'uscita modem, quella contrassegnata con il simbolo del telefono. Poi è necessario entrare nel pannello di controllo (come spieghiamo nel prossimo capitolo) e dal menù slot selezionare la voce Slot 7: Built-in Apple Talk. Apple IIgs segnala che "WARNING: Printer - Modem - Apple Talk cannot be active at the same time!!!" cioè che non è possibile usare contemporaneamen-

te una stampante, un modem e la rete, per il semplice motivo che esistono fisicamente sul retro della macchina connettori per due di queste tre funzioni. È dunque necessario disinserire lo Slot 1: Printer Port o lo Slot 2: Modem Port.

Se nella rete sono a disposizione più stampanti, il meccanismo di rete attraverso il network driver vi chiede di sceglierne una, e vi comunica se tutte le stampanti sono occupate e dovete attendere. È possibile, naturalmente, scambiare messaggi e documenti attraverso la rete.

1.4 I dati tecnici

Apple IIgs è basato su un processore a sedici bit, cioè può trattare numeri interi compresi tra zero e 65.535 in un'unica operazione.

Il clock del processore è di 2,8 MhZ, cioè 2.800.000 microoperazioni al secondo: le operazioni del linguaggio macchina richiedono tra due e sette cicli di clock per venire eseguiti. Tutte le operazioni macchina hanno un codice di un solo byte. Nell'eseguire i programmi utente (in RAM) il processore è rallentato, così come avviene in tutti i computer, a 2,6 MhZ dai cicli di refresh, mentre il codice in ROM viaggia alla velocità massima di 2,8 MhZ.

Il bus dati è ad otto bit e il bus degli indirizzi a ventiquattro bit: 16 Mbyte in tutto. I sedici bit bassi del bus indirizzi sono disponibili agli slot di espansione per le periferiche.

Il collegamento per monitor è disponibile su retro anche per colore secondo il sistema RGB, analogico per consentire una migliore trasmissione dei colori, e richiede una banda passante del monitor di almeno 16 MhZ.

I precedenti modelli di Apple II usavano una uscita digitale, e quindi non è possibile usare gli stessi monitor a colori per le due macchine, mentre i monitor in bianco e nero funzionano.

Le periferiche di input (tastiera, mouse, tavoletta grafica, rollerball eccetera) sono connesse tramite l'Apple DeskTop Bus che permette di usare sino a sedici differenti periferiche collegate in serie, anche dello stesso tipo, distinguendole e servendole secondo un protocollo intelligente grazie ad un processore dedicato.

L'orologio calendario è realizzato con un custom chip appositamente realizzato in tecnologia CMOS ed è alimentato da una micropila al litio. Viene consultato da programma inoltrando una richiesta al sistema operativo ProDOS.

Il generatore di suono è un chip sintetizzatore Ensoniq DOC (digital oscillator chip) a trentadue oscillatori con 64 Kbyte dedicati che genera 15 voci stereofoniche contemporaneamente, a qualunque forma d'onda, ed è in grado di sintetizzare la voce. È possibile utilizzarlo per campionare un suono in input e riprodurlo in seguito, memorizzandolo in RAM, usando un convertitore analogico-digitale ADC; il segnale va condizionato usando un filtro con cutoff inferiore ai 14 kHz o aggiungendo un circuito di ritenzione sincronizzato al clock del chip. Un oscillatore è sempre disponibile come clock per il chip, un secondo è lasciato disponibile per future modifiche al sistema. È possibile collegare il computer con un complesso stereofonico tramite la presa minijack–L stereofonica sul retro della macchina.

Il computer dispone di un trasformatore incorporato. Riceve corrente alternata tra il 210 e i 240 volt e indifferentemente a 50 o 60 hertz. L'assorbimento è normalmente di soli 11 Watt (60 Watt nominali). Temperatura ambiente tra gli zero e i 45 gradi centigradi, umidità tra 5% e 85%.

2

DAL PANNELLO DI CONTROLLO

L'innovazione più palese nell'Apple IIgs è il pannello di controllo, un programma permanentemente residente nel computer (è registrato nella memoria permanente ROM) che permette di modificare lo status di Phoenix secondo le esigenze del momento.

L'uso del pannello di controllo è dettagliato nell'Appendice A del Manuale Utente che vi è stato fornito insieme al computer. In questo capitolo correggiamo alcune inesattezze del Manuale Utente, aggiungiamo considerazioni e trucchi utili, infine ci dedichiamo a spiegare cos'è realmente e come può funzionare il pannello. Ma per prima cosa, se ancora non lo avete fatto, leggete il Manuale Utente! È l'ultimo avvertimento.

2.1 Gli Accessori di Scrivania

Secondo la filosofia Apple, dovreste pensare al vostro computer come ad una scrivania. Sulla scrivania stanno i volumi (i dischetti) che contengono le applicazioni (i programmi) ed i dati.

Esistono poi gli Accessori di Scrivania, dei sottoprogrammi di varia utilità che potete utilizzare da dentro un programma: per esempio, poniamo che il sottoscritto mentre scriva il paragrafo sulle espansioni di memoria debba sapere quanti byte di memoria libera ci sono in un Apple IIgs: sono 256 * 1024, cioè...

Sto usando un calcolatore e non posso neppure usarlo per fare una semplicissima moltiplicazione?

Ecco dove arrivano gli accessori di scrivania. Una calcolatrice è un esempio tipico di accessorio di scrivania che grazie ad Apple IIgs ho sempre a disposizione, anche mentre sto usando un elaboratore di testi che non prevede questa necessità.

Gli accessori di scrivania sono stati introdotti per la prima volta su Apple Lisa e sul suo successore Macintosh: hanno fatto poi capolino su Apple II, ma solo i programmi dell'ultima generazione potevano disporne, e spesso un accessorio funzionava solo su alcuni programmi e non su altri.

Apple IIgs dispone di due tipi differenti di accessori di scrivania, chiamati **classici** e **nuovi**. Gli accessori classici sono utilizzati tramite il Pannello di Controllo, quelli nuovi sono disponibili solo nel menù mela delle applicazioni create appositamente per Phoenix.

Dal punto di vista del programmatore, si tratta di programmi che vengono eseguiti dal sistema operativo togliendo il controllo all'applicazione presente e che al termine restituiscono il controllo all'applicazione senza che questa ne sia conscia. Gli accessori nuovi usano lo schermo grafico di super alta risoluzione, aprendo una finestra per l'output e rilasciandola quando hanno terminato l'esecuzione. Gli accessori nuovi possono utilizzare una forma semplificata di multiprogrammazione se l'applicazione concorrente usa il metodo normale di gestione dell'input (il Ciclo Eventi), grazie all'uso degli interrupt.

Occupiamoci innanzitutto degli accessori classici: si ottiene un menu degli accessori disponibili da qualunque programma premendo Control-Mela Vuota-Esc (i tre tasti insieme). Appare la scritta **Desk Accessories** e potete scegliere tra le voci Control Panel, Alternate Display Mode e Quit (e probabilmente altre ancora).

Del pannello di controllo ci occupiamo nel prossimo paragrafo. Dell'Alternate Display Mode parleremo nel prossimo capitolo. Con Quit tornate al programma che avevate lasciato, al punto in cui lo avevate lasciato.

È meglio non premere Control-Mela Vuota-Esc mentre la spia rossa dei disk drive è accesa. Può darsi che così facendo interrompiate un programma mentre sta leggendo dati dal disco, il che lascerebbe il disco a girare in attesa di ordini dal programma mentre il controllo della macchina è passato a voi ed il program-

ma è temporaneamente inattivo. Nulla di grave, se accade, ma è buona abitudine evitarlo per non logorare (di un poco) il dischetto.

Se il programma che si vuole interrompere per usare un accessorio è basato sul sistema operativo ProDOS in versione 1.0, 1.0.1, 1.0.2, 1.1.1 (lo potete sapere leggendo la scritta che appare per prima lanciando il dischetto) non sarete normalmente in grado di invocare il menù degli Accessori (alla fine del capitolo avremo scoperto il perché).

Si può facilmente risolvere il problema: basta copiare il nuovo ProDOS sul vecchio disco. Prendete il disco "Mouse Desk 2.0" che vi è stato fornito insieme al computer e copiate il file /MOUSE.DESK/SYSTEM/P8 sul vecchio disco, dandogli il nome PRODOS. C'è già un file con quel nome sul vecchio disco: cancellatelo pure.

Gli Accessori di Scrivania classici disponibili sono il pannello di controllo e le altre voci che appaiono nel menu. Apple IIgs inserisce in memoria RAM e presenta nel menu gli accessori di scrivania che trova sul disco di lancio dentro /nomedisco/SYSTEM/DESK.ACCS mentre, come dicevamo, il pannello è sempre disponibile.

Il numero di accessori di scrivania che possiamo avere a disposizione contemporaneamente è limitato solo dalla memoria RAM che Apple IIgs ha in sé.

2.2 Il Pannello

Nel pannello di controllo voi configurate il vostro Apple IIgs in modo da usarlo il più comodamente possibile.

Le indicazioni del pannello possono essere "lette" dalle applicazioni create più di recente che in questo modo possono adattarsi ai nostri capricci. Quaranta od ottanta colonne? Colore o bianco e nero?

Le applicazioni create prima di Apple IIgs, ovviamente, non sanno nulla dell'esistenza del pannello. Cosa succede allora delle sue indicazioni?

Alcune delle preferenze espresse via pannello non possono venire toccate dalle vecchie applicazioni; sono:

* Il colore del testo, dello sfondo e del bordo
* La velocità
* Il volume del suono e la nota del beep
* Tutto quanto specificato nel menù Options
* La visibilità delle periferiche specificata nel menù Slot
* La capacità del disco RAM

Altre caratteristiche possono venire modificate a proprio piacere dalle applicazioni (qualche volta con nostra irritazione). Sono:

* I parametri della Stampante e del Modem
* La disposizione del testo in 40 od 80 colonne
* La possibilità di usare il disco RAM
* La possibilità di conoscere data ed ora dall'orologio

Qualche voce del pannello merita una spiegazione dettagliata.

Sotto la voce **Display** trovate la possibilità di selezionare il **Tipo** come colore o bianco e nero: si tratta di una scelta più delicata di quanto possa sembrare.

I monitor a colori non RGB e (molto di più) le televisioni a colori hanno la tendenza a "sbavare": quelli che su un monitor bianco e nero sono caratteri nitidissimi e perfettamente leggibili su una televisione divengono spesso opachi e sfrangiati di molti colori. Per questo motivo nell'Apple II esiste (ed è sempre esistito) un circuito chiamato *color killer*, uccisore del colore. Il circuito costringe la televisione a visualizzare i caratteri in bianco e nero senza problemi.

Nei primi Apple II il color killer era automatizzato, ed entrava in azione quando il computer veniva disposto in modo testo, mentre veniva disinserito quando si scegliesse l'alta risoluzione grafica. Con Apple IIgs si può scegliere se il color killer va mantenuto in vita con la doppia alta risoluzione abilitata: molti programmi (tra cui Mouse Desk e Instant Pascal, distribuiti dalla Apple) fanno uso della doppia alta risoluzione, il modo grafico più potente tra quelli disponibili anche ai vecchi modelli di Apple II, per visualizzare del testo, ma con un monitor a colori le scritte risultano illeggibili; da qui la scelta di rendere operante il color killer su richiesta anche in quel momento.

Il color killer, a prescindere dalla vostra scelta da pannello, è inattivo con la super alta risoluzione tipica dell'Apple IIgs che per la sua natura non provoca sbavature, e con la normale alta risoluzione (in quest'ultimo caso parrebbe una scelta sbagliata). Il

Pannello di Controllo va sempre selezionato su Monochrome se il vostro monitor è monocromatico, e trasforma il segnale del colore in tinte di grigio.

Nel menù **Options** le prime due voci permettono di selezionare le caratteristiche della tastiera.

Le tastiere delle macchine da scrivere, dalle quali si è preso lo spunto nel disegnare le tastiere dei computer, sono differenti in ogni Paese: quella italiana deve contenere le vocali accentate, quella svedese non può fare a meno della A sormontata da un circolo, quella danese abbisogna delle O tagliate, quella francese della cediglia, la lettera ç, e così via. Apple IIgs è predisposto per poter lavorare con otto tastiere differenti, compresa quella italiana, a differenza del suo Grande Rivale che, spocchioso, si limita ad imporre ovunque la scomodissima tastiera americana.

La tastiera italiana è conosciuta con il nimignolo di "qzerty": deriva dall'elenco delle prime sei lettere che si trovano in alto a sinistra. La tastiera americana è la *qwerty*, quella francese la *azerty* e così via.

Il problema delle tastiere, però, non si risolve tanto facilmente sostituendo il pezzo.

Per un computer, una lettera non significa nulla: al livello più interno, la macchina sa trattare unicamente con i numeri. Si segue allora il principio di associare ad ogni lettera, cifra e simbolo speciale (punteggiatura, parentesi e simili) un numero intero tra lo 0 e il 127, secondo una convenzione chiamata **codice ASCII**. Si pronuncia "aski", e significa *american standard code for information interchange*, cioè "codice standard americano per lo scambio di informazioni".

Ed ecco l'inghippo: come dice il suo nome, ASCII è un codice americano, che non prevede simboli nazionali come le nostre vocali accentate o la O tagliata o la ç. Delle due soluzioni possibili, cambiare il codice ASCII oppure sostituire alcuni simboli poco usati con quelli nazionali, i costruttori di Apple IIgs hanno scelto il secondo (su Macintosh è stata preferita la prima possibilità).

Esistono così molte versioni leggermente differenti del codice ASCII, i cosiddetti **codici ASCII nazionalizzati**. Nella tabella 2.1 potete trovare le differenze tra gli otto codici nazionali più importanti, quelli che Phoenix riconosce.

Per il computer non esiste alcuna differenza, per esempio, tra la i accentata e la tilde (quella specie di esse ruotata di novanta gradi): poiché entrambe hanno lo stesso numero di corrispon-

denza, per il computer i due caratteri sono identici: è il generatore di caratteri, quel piccolo e trascurato componente che associa ai numeri trasmessi dal computer le lettere e i simboli, che vuole sapere a quale convenzione nazionale ci vogliamo riferire, ed a lui solo arriva l'indicazione selezionata con **Display Language**. Per lo stesso motivo, quando una scritta appare sullo schermo grafico, potreste avere la sorpresa di trovare una tilde quando avevate premuto il tasto con la i accentata. Questo significa solamente che state usando un programma creato per il mercato americano; i caratteri grafici sono generati direttamente dal programma, e non passano per il generatore di caratteri, dunque vi ritrovate i simboli americani.

D'altra parte, Apple IIgs permette di usare otto diverse tastiere nazionali: questo significa che il chip schiavo della tastiera deve sapere se voi premendo il secondo tasto lettera da sinistra intendete riferirvi alla zeta di qZerty o alla doppia vi di qWerty. Al chip di tastiera ed a lui soltanto vanno le indicazioni che stabilite con **Keyboard Layout.**

Normalmente, sia Keyboard Layout che Display Language dovrebbero indicare Italian, cioè segnalare che state usando una tastiera italiana per ottenere scritte in italiano. Ma se siete un programmatore avrete bisogno dei simboli americani, come le parentesi graffe per il Pascal o il linguaggio C: scegliete allora Display Language: U.S.A.

D'altra parte un videogioco può lasciarvi guidare il vostro personaggio sullo schermo usando la tastiera: una convenzione tipica è usare i tasti <– e –> ,che troverete in basso a destra, per muovervi orizzontalmente, e i tasti A e Z per muoversi in alto e basso. Disgrazia vuole che sulla tastiera americana la Z stia sotto la A, mentre in quella italiana vale il viceversa: e non vi sembrerà certo naturale dover premere il tasto più in alto per far scendere il vostro omino e viceversa. Basterà per risolvere il dilemma scegliere Keyboard Layout: U.S.A. e potrete usare il tasto A per salire e W per scendere: anche se voi batterete il tasto W il chip di tastiera, istruito dal pannello di controllo, farà arrivare al programma una zeta, permettendovi di vivere felici.

Un'altra convenzione molto usata è il tasto I per salire, J per la sinistra, K per la destra, M per il basso: ma nella tastiera italiana la M non si trova, come in quella americana, nell'ultima fila di tasti. Modificare il Keyboard Layout risolve tutti questi problemi; ma ricordatevi di rimettere le cose come stavano, o avrete dei bruttissimi scherzi riprendendo ad usare Phoenix per battere una lettera.

2.3 Il disco RAM

Da Pannello di Controllo possiamo scegliere le dimensioni del disco RAM: questa opzione è disponibile solo sugli Apple IIgs con almeno 512 Kbyte di memoria RAM. Il disco RAM è gestito direttamente dal Memory Manager in collaborazione con il sistema operativo, ed è utilizzabile con il sistema operativo ProDOS o con UCSD Pascal dalla versione 1.3 in poi.

Non bisogna confondere il disco RAM dell'Apple IIgs con il corrispondente esistente negli Apple IIe e IIc o in Apple Macintosh. Il disco RAM di Phoenix, a differenza degli altri, conserva i file contenuti anche se rilanciate la macchina con un Control-Mela Vuota-Reset, se cambiate sistema operativo o se ne cambiate le dimensioni da Pannello di Controllo; le informazioni vanno perse se e solo se spegnete la macchina.

Usando il Basic Applesoft sotto ProDOS 8 vi ritroverete a disporre di due distinti dischi RAM, chiamati /RAM e /RAM5: il primo è lo stesso disco RAM dei precedenti modelli con 128 Kbyte, e può venire cancellato senza preavviso da qualunque applicazione faccia uso di 128 Kbyte (come AppleWorks, DazzleDraw, AppleWriter, Istant Pascal, Pinpoint eccetera) oppure rilanciando la macchina con Control-Mela Vuota-Reset; il secondo è il disco RAM particolare di Phoenix che abbiamo appena descritto. I due dischi sono visti rispettivamente come Slot 3 Drive 2 e come Slot 5 Drive 2.

2.4 Il Pannello e la Stampante

Da Pannello di controllo è possibile guidare le interfacce di comunicazione RS422, ovvero scegliere i parametri con i quali Apple IIgs comunica con la stampante e il modem: sono stati raggruppati alle voci Printer Port e Modem Port del pannello.

Nei modelli sino all'Apple IIe le schede di comunicazione erano degli optional da inserire negli slot della macchina, ed i parametri venivano scelti muovendo dei piccolissimi interruttori che si trovavano sulla scheda, chiamati i **dip switch**. Con l'Apple IIc è stato scelto di rendere standard la dotazione delle schede di comunicazione: nel IIc le schede sono incorporate, e i parametri

vengono selezionati inviando alla scheda, prima del messaggio vero e proprio da trasmettere alla stampante o al modem, dei codici speciali di controllo.

Apple IIgs ha due schede incorporate come il IIc, ed i parametri *default*, cioè standard, sotto i quali operano le schede sono gli stessi: è però possibile cambiarli da pannello di controllo oltre che inviando quei codici. Per questo motivo, i programmi creati per Apple IIc non tengono in considerazione i parametri indicati nel Pannello, perché ne ignorano l'esistenza.

Normalmente i parametri predisposti nel Pannello e indicati con il segno di spuntatura, i default, sono i più adatti a tutti gli scopi; le stampanti Apple, ad esempio, sono costruite per rispondere a quelle specifiche.

Una possibilità interessante da segnalare è quella di selezionare "Buffering: Yes": in quel modo quando la lenta stampante non può tenere dietro al veloce computer che tenta di trasmettere un testo più velocemente di quanto la stampante possa stampare, viene riservato dentro Apple IIgs un buffer, cioè uno spazio di memoria usato dalla stampante per conservare i dati in attesa di stampa: questo evita al computer di dover aspettare che la stampante abbia terminato le stampe per procedere, e consente di utilizzare contemporaneamente il computer e la stampante.

Con alcune stampanti non Apple potrebbe presentarsi una difficoltà facilmente aggirabile grazie al Pannello. Se la stampante lascia regolarmente una riga bianca di spazio indesiderata tra ogni riga a stampa, sarà sufficiente modificare il valore di "Add LF after CR" da Yes a No.

Il chip Zilog che controlla ciascuna delle interfacce seriali è particolarmente potente: una delle caratteristiche più interessanti è la capacità di gestire la cosiddetta "stampa in background". Questo significa che, dedicando una parte della memoria interna di Apple IIgs (compresa tra un minimo di 128 byte e un massimo di 64 Kbyte) al processore Zilog è possibile fargli governare la stampante senza intervento da parte del processore 65816. In pratica, questo si traduce nella capacità di Apple IIgs di continuare a lavorare mentre si sta eseguendo una stampa, mentre tutti gli altri computer sono costretti a perdere il loro tempo servendo la stampante, e l'utente del computer si ritrova con il computer inutilizzabile per tutto il tempo necessario a completare la stampa.

2.5 Come aggiungere accessori di scrivania

Gli accessori di scrivania che volete avere sempre a disposizione debbono venire ricopiati nella subdirectory SYSTEM/DESK.ACCS del disco che introducete accendendo la macchina. Gli accessori classici sono contenuti nei file indicati con il tipo $B5 e quelli nuovi hanno il tipo $B6 (ad entrambi i codici numerici verrà certamente associato un nome convenzionale, che per il momento però non esiste).

Nella subdirectory SYSTEM si trova anche la subdirectory TOOLS che contiene eventuali tool da affiancare a quelli in ROM del Toolbox (vedi capitolo 9) o magari da sostituire, nel caso che venga sviluppata una versione più recente, più veloce o più eclettica. Il Tool Locator è in grado di far fronte a questa situazione. Nella subdirectory DRIVERS trovano posto i device driver, nella subdirectory FONTS stanno le fonti di caratteri per la super alta risoluzione: vi basta copiare in quella subdirectory altri file del genere per vedere automaticamente allungata la lista di stili di scrittura a vostra disposizione.

Il sottoscritto mantiene un dischetto senza programmi con il solo scopo in vita di contenere tutti i tool, gli accessori, i driver più aggiornati. Accendendo il mio computer effettuo il bootstrap con quel disco, e da quello passo poi ad eseguire il programma che mi interessa. Potrebbe essere un consiglio da imitare.

Per i programmatori: gli accessori possono fare richieste al sistema operativo ProDOS 16 ed ai manager, il cui codice è in buona parte rientrante. Se viene inoltrata una richiesta ad un tool non rientrante lo Scheduler mette il programma in sospensione in una coda di attesa sul tool. Il sistema di sviluppo Apple permette di scrivere accessori in Pascal ed in linguaggio C oltre che in Assembly 65816, e nel package sono forniti degli esempi funzionanti con tanto di codice sorgente.

3

I PROGRAMMI

Solo alcune brevi note sui programmi più diffusi con Apple IIgs, per proseguire illustrando in breve l'uso del sistema di sviluppo software, lo Apple Programmer's Workshop. Infine, ci occuperemo dei programmi creati per tutta la serie II: dei problemi di compatibilità e di come risolverli, di come sfruttare le caratteristiche di Phoenix per migliorarne le capacità.

3.1 Finder, Paint e Write

Esistono due finder per Apple IIgs, cioè due programmi che permettono di visualizzare i file del disco in forma iconica, di copiare dischi e files, di chiedere informazioni sul disco eccetera. Il primo ad entrare in circolazione è stato Mouse Desk 2.0 della Version Soft (distribuito in Italia dalla Italware), il secondo è stato il Finder della Apple Computer.

Mouse Desk non è un programma creato esclusivamente per Apple IIgs, e funziona altrettanto bene sui modelli IIe e IIc: inoltre, essendo stato distribuito il più presto possibile dai suoi creatori, non è stato perfettamente provato e la versione 2.0 contiene diversi errori che saranno rimediati al più presto. Il Finder Apple è stato invece creato esplicitamente per funzionare solo su Apple IIgs sotto ProDOS-16.

GS Paint, conosciuto anche con il nome provvisorio di Paint-

Works, è stato anch'esso creato dalla Version Soft. È un eccellente programma, eccetto per un paio di difetti sui quali voglio attirare l'attenzione del mio lettore.

Se viene scelta una tavolozza composta di un solo colore ripetuto sedici volte, il programma accetta l'imposizione, e lo schermo diviene uniformemente di quel colore con l'unica eccezione della mela in alto a sinistra. Questo potrebbe essere pericoloso perché potrebbe far perdere parte del lavoro compiuto.

Una caratteristica molto brillante di GS Paint va commentata, perché serve a comprendere come funzioni la gestione del colore sull'Apple IIgs. Se battete due volte il pulsante del mouse su un colore della tavolozza vi è possibile editare i colori scelti in modo da accedere a tutta la gamma di 4.096 colori puri disponibili al computer. Noterete sulla sinistra tre sfogliatori colorati e sulla destra altri due un poco isolati. All'estrema sinistra un rettangolo colorato e sotto di questo un altro diviso in sedici, che rappresenta i sedici colori della tavolozza standard.

Apple IIgs forma i colori come combinazione dei tre colori fondamentali: rosso verde e blu, cui corrispondono i tre sfogliatori di sinistra nell'ordine. Un colore è aumentato al massimo abbassando al massimo lo sforglitore relativo: provate ad abbassare il primo ed alzare gli altri due ed otterrete il rosso puro. Annullando un colore ed abbassando gli sfogliatori degli altri due trovate il colore complementare: se alzate al massimo il cursore del primo sfogliatore ed abbassate al minimo gli altri due otterrete l'azzurro purissimo, dal secondo il cyan (una specie di lillà) e dal terzo il giallo. Ovviamente, combinando tutti e tre i colori avrete il bianco ed azzerandoli ambetré otterrete il nero. Provate a giocare con GS Paint per qualche tempo per rinverdire i ricordi di teoria dei colori delle scuole medie inferiori.

Ciascuno dei tre sfogliatori può assumere sedici posizioni (provate a contarle facendo click sulle frecce). Dato che i tre sfogliatori sono indipendenti otteniamo 16 * 16 * 16 = 4.096 colori differenti.

Quando un programma vuole selezionare un colore tra i 4.096 del calcolatore lo fa con un numero di dodici bit, o se preferite un byte e mezzo: quattro bit (che rappresentano appunto un numero tra uno e sedici) sono dedicati a ciascuno dei colori fondamentali. Se è necessario un rosso vivo, si scrive 1111 0000 0000 ovvero in esadecimale $F00; per il giallo avremo 1111 1111 0000 ovvero $FF0; al nero corrisponde lo zero (assenza totale di colore).

3.2 L'ambiente di sviluppo

La Apple ha messo a disposizione un ambiente di sviluppo software, creato da una casa indipendente, e tre linguaggi di programmazione sviluppati da altrettante software house (Assembler, Orca, C MicroMax, Pascal) per creare programmi che sfruttino al massimo le capacità del suo ultimo nato.
Uno scopo dichiarato è di permettere di scrivere contemporaneamente programmi sia per Apple IIgs che per Macintosh: i linguaggi, il toolbox ed il sistema operativo delle due macchine sono stati resi il più possibile simili.
Come è ormai consueto per Apple, la documentazione è piuttosto abbondante ed onnicomprensiva, ed ha il solo difetto di essere difficile da trovare in Italia. Bisogna inoltre fare attenzione, perché poco prima dell'annuncio ufficiale della nascita di Phoenix è stata distribuita dalla Apple Computer agli sviluppatori di software una versione non definitiva della documentazione tecnica: affidarsi a quei documenti, magari scambiandoli per ufficiali, può significare andare incontro ad errori disastrosi nel caso di modifiche dell'ultimo minuto alla macchina. I manuali inaffidabili sono contrassegnati dalla scritta "draft".

Altri linguaggi sono stati scritti e verranno scritti per Apple IIgs, oltre a quelli ufficiali della Apple Computers: la struttura interna del microprocessore 65C816 è tale da rendere semplice la stesura di compilatori e linguaggi in generale, tanto quanto era difficile sul vecchio 65C202. Per non dilungarci troppo, tratteremo qui solo dello Apple Programmer's Workshop.

L'ambiente nativo è uno shell molto simile a quello del sistema operativo UNIX, nel quale si possono eseguire programmi, dare comandi eseguiti dallo shell e scrivere semplici programmi nel linguaggio di shell che eseguano combinazioni di comandi e programmi. Nell'insieme lo shell è piuttosto comodo da usare, potente e flessibile una volta che ci si è abituati: ricorda oltre a Unix anche certe caratteristiche di MS-DOS e dell'ambiente Basic sotto ProDOS 8, il Basic.System, comune a tutti i programmatori Apple. Sono disponibili i file eseguibili dal linguaggio di shell che nella terminologia Apple si chiamano file EXEC, e nel linguaggio Unix andrebbero chiamati gli Script), ed è possibile passare parametri alle applicazioni attivandole - i parametri argc ed argv del linguaggio C.

Sarebbe buona cosa che i futuri linguaggi creati per Apple IIgs siano in grado di funzionare dentro l'Apple Workshop; questo costringerebbe tutti i programmatori a procurarsi il Workshop oltre al linguaggio scelto, ma permetterebbe di scrivere programmi in più linguaggi, sviluppando i blocchi funzionali nel linguaggio più appropriato (blocchi matematici in Fortran, punti in cui sono necessari i tipi di dati astratti in Pascal, procedure in cui è richiesta una alta velocità in Assembler o magari linguaggio C, eccetera). Il linker fornito con lo Apple Programmer's Workshop è infatti in grado di risolvere le unioni tra linguaggi diversi.

Qualcuno obbietterà di fronte alla veste spartana del Workshop, preferendo un ambiente di sviluppo più impressionante e amichevole, meno mnemonico anche se più lento, tipo Macintosh. Personalmente ho sempre sostenuto, e ripeto anche qui, che l'interfaccia utente a finestre è una grande invenzione ma non sempre la più adatta, e che per la programmazione uno shell con script e macroistruzioni è il più adatto per i programmatori professionisti e in generale "seri", anche se disturba principianti e programmatori della domenica.
Uno dei manuali Apple di corredo ad Apple IIgs documenta nei dettagli caratteristiche e capacità del Workshop.

3.3 Apple IIgs e i programmi creati per i modelli precedenti

Apple IIgs tende a comportarsi come un Apple IIe enhanced con 128 Kbyte di RAM e sette schede negli slot quando viene usato da un programma uscito nel 1986 o prima. Le sue capacità superiori sono mascherate, ed un gran lavorio, chiamato in gergo *shadowing,* è compiuto dai chip interni per simulare un vecchio modello.

Se il programma in questione non è un videogioco, può essere una buona scelta aumentare di due volte e mezza i tempi di elaborazione e di attesa scegliendo l'alta velocità. Ben poco d'altro può essere fatto per far sfruttare meglio al programma le capacità aggiuntive della macchina: una eccezione è un programma scritto in Pascal del quale si dispone del codice sorgente. In quel caso sarà possibile trasferire il sorgente dal sistema UCSD a

ProDOS 16 (con un programma come Universal File Converter) e poi ricompilarlo usando il Pascal di Apple IIgs.

Per aumentare ancora la velocità di elaborazione di un programma in modo emulazione è possibile chiedere ad Apple IIgs di ridurre i tempi di shadowing, dichiarando quali display grafici non vengono usati dal programma.

In modo emulazione gli spazi dei display di testo e di grafica vengono continuamente ricopiati dal banco zero al banco $E0 (si veda il capitolo 9), ma gran parte di questo lavoro è normalmente inutile, perché nessun programma usa sia il testo che alta, bassa e doppia alta risoluzione.

È sufficiente toccare il cosiddetto Shadow Register alla locazione $C035, i cui bit hanno il significato spiegato nella tabella seguente:

Bit	Funzione
7	Riservato (non significativo, ma è necessario scrivere 0)
6	Shadowing delle locazioni di I/O e della language card
5	Riservato (non significativo, ma è necessario scrivere 0)
4	Shadowing della doppia alta risoluzione
3	Shadowing della super alta risoluzione
2	Shadowing della alta risoluzione, pagina 2
1	Shadowing della alta risoluzione, pagina 1
0	Shadowing della pagina di testo (ottanta colonne)

Scrivere il bit = 1 inibisce lo shadowing, velocizzando i tempi. Le locazioni di I/O andrebero sempre lasciate in shadow e quelle di super alta risoluzione mai. Lo shadowing può essere controllato con lo pseudo registro Quagmire di System Monitor (vedi capitolo 8).

3.4 Come risolvere i problemi di compatibilità

Un certo numero di programmi creato prima dell'uscita dell'Apple IIgs ha problemi a funzionare correttamente sulla nuova

macchina. Prima di rinunciare ad usare il vecchio software, andrebbe tentato un certo numero di accorgimenti che spesso risolve la situazione.

Da pannello di controllo selezionate: System Speed: Normal; Display - Columns: 40; Slots - Startup Slot: Scan; Options: tutte le voci sul default indicato dal segno di spuntatura, eccetto le prime due voci che sono irrilevanti e che potete lasciare al loro valore normale.
Dal menù degli accessori di scrivania classici scegliete Alternate Display Mode e confermate la scelta.
A questo punto uscite dal pannello di controllo e tentate di far partire il programma incriminato con un Control-Mela Vuota-Reset.

Tra i programmi più famosi, ho trovato che solo Apple Writer IIe ed il copiatore Locksmith nella versione 6.0 non funzionano su Apple IIgs usando questi accorgimenti per aumentare la compatibilità con il vecchio software.

Qualche volta accade che un vecchio programma sembra disinserire la tastiera, e non accetta più nessun comando che ne proviene. Quasi sempre, per rimediare questo problema è sufficiente premere Control-Mela Vuota-Delete.

Numerosissimi vecchi programmi hanno problemi ad effettuare stampe una volta installati su Apple IIgs, nonostante funzionino perfettamente sotto ogni altro aspetto. Questo dipende dal fatto che l'integrato ACIA 6551 che regolava le comunicazioni sugli Apple IIe e IIc è stato sostituito con il più potente Intel SCC 8530: i programmi che comunicavano direttamente con il processore 6551 (cosa da non farsi secondo le convenzioni pubblicate dalla Apple, ma ciononostante una abitudine molto diffusa) si ritrovano un processore diverso e si bloccano.
Se avete assoluta necessità di usare un vecchio programma per stampare usando Apple IIgs l'unica scelta è di acquistare una Super Serial Card, inserirla nello slot 1, selezionare da pannello di controllo Slots - Slot 1: Your Card e vivere felici.

Un ultimo inconveniente, molto raro: se un programma nella fase di lancio fà apparire prima una parentesi quadra chiusa, poi una serie di comandi Monitor (un asterisco seguito da numeri in esadecimale), ed infine si blocca, significa che stava usando un file EXEC per agire sulla language card (vedi capitolo 9 paragrafo 1). Apple IIgs, per evitare il blocco del sistema, impedisce la modifica manuale della language card. In questo caso è necessario

modificare il codice del programma, usando un caricatore di language card analogo a quello usato nel System Master DOS 3.3.

Grazie allo Slot Maker di cui abbiamo parlato nel primo capitolo, la compatibilità con le schede di interfaccia sviluppate per Apple IIe è quasi totale: se avete però schede particolari, magari create su misura per scopi speciali, potrebbero sorgere problemi. Non esiste un principio generale, e l'unica soluzione è quella pragmatica di tentare la scheda su Apple IIgs: comunque, non dovrebbe esserci nessuna possibilità di danneggiarlo sinché si seguono i soliti accorgimenti. L'unica scheda "ufficiale" ad avere problemi su Apple IIgs è quella per il ProFile nelle prime versioni, come spiegheremo nel capitolo 6.

4

MUSIC AND LIGHTS

Non si chiama Apple IIgs per nulla: le frecce in più al suo arco sono la Grafica ed il Suono, ed in questo capitolo vedremo esattamente cosa Phoenix può fare in questi campi. Per completezza, il capitolo inizia illustrando i modi grafici già disponibili con i precedenti modelli di Apple II, e prosegue poi con la super alta risoluzione di Apple IIgs. Per ultimo viene il generatore di suoni.

Due precisazioni prima di cominciare.
Apple II dispone di uno schermo di solo testo, di ventiquattro righe per quaranta od ottanta colonne, in alternativa ai modi grafici. Nel modo testo lettere e cifre possono apparire solo nei punti prestabiliti della griglia, ed è possibile visualizzare solo i simboli standard dell'alfabeto: per questo motivo spesso si fa uso degli schermi grafici per la visualizzazione di testo, disegnando i caratteri sullo schermo grafico. Alcuni computer, per esempio il fratellone Apple Macintosh, non dispongono di un modo testo, ed utilizzano esclusivamente lo schermo grafico: scelta rispettabilissima, che ha però un tallone d'Achille: uno schermo di testo permette la visualizzazione rapidissima delle scritte, uno schermo grafico richiede molte più operazioni ed è perciò più lento di qualche ordine di grandezza.

Nel parlare di grafici sui personal computer si fa normalmente una grande confusione. Uno degli aspetti da prendere in considerazione è, certo, la quantità di **pixel** (termine tecnico che deriva da "picture element", elemento di immagine, e che indica la più piccola porzione dello schermo selezionabile) disponibile,

ma altrettanto importanti sono altri elementi. La quantità di colori disponibili in tutto (la cosiddetta palette o tavolozza) è fondamentale, così come bisognerebbe dichiarare quanto ampio sia il sottoinsieme di questi visualizzabile contemporaneamente; se vi siano eventuali limitazioni (per esempio, limitazioni sul posizionamento dei colori sullo schermo) e la complessità del bit mapping. Con quest'ultimo termine indichiamo il modo in cui vengono fatti corrispondere i byte in memoria e i pixel sullo schermo: più semplice e lineare è il bit mapping, più semplice è l'uso della grafica.

Infine, vanno dichiarate le capacità hardware del computer nell'animazione: cioè le potenzialità intrinseche del microprocessore e quelle degli eventuali processori schiavi.

Per quanto riguarda Apple IIgs, va detto che il processore è piuttosto ben dotato, ma non esistono processori schiavi che lo aiutino nella gestione della grafica come si trovano normalmente, per esempio, sui personal della Commodore. Una caratteristica interessante riguarda tutti i modi grafici tranne la super alta risoluzione: sono disponibili due **pagine grafiche**, cioè si possono gestire indipendentemente due schermi. Questo è un particolare molto interessante per gli sviluppatori di software in quanto permette ottime animazioni grafiche.

Tutti i modi grafici, ancora una volta ad eccezione della super alta risoluzione, possono innescare anziché lo schermo intero solo i cinque sesti più in alto dello schermo per il loro display, lasciando lo spazio per quattro righe di testo nella parte inferiore dello schermo. In questo modo, ovviamente, la risoluzione verticale viene diminuita; ad esempio nell'alta risoluzione scende da 192 a 160 pixel (gli altri sono coperti dal display di testo).

4.1 I modi grafici della serie II

Tutti i modelli di Apple II, sino dal primo con soli 16 Kbyte di memoria, possono visualizzare grafici in sedici colori. Il primo modo grafico disponibile sull'Apple II li può usare tutti contemporaneamente, senza limitazioni. I sedici colori sono elencati nella tabella 1; elenco anche i nomi in inglese per quanti volessero criticare le traduzioni della Apple Italia.

Tabella. I colori di bassa risoluzione dell'Apple II			
Black	Nero	Brown	Marrone
Magenta	Magenta	Orange	Arancione
Dark Blue	Blu scuro	Grey 2	Grigio scuro
Violet	Porpora	Pink	Rosa
Dark Green	Verde scuro	Bright Green	Verde
Grey 1	Grigio chiaro	Yellow	Giallo
Medium Blue	Blu	Aqua	Acqua
Light Blue	Blu elettrico	White	Bianco
I colori di alta risoluzione dell'Apple II			
Black1	Nero1	Black2	Nero2
Green	Verde	Orange	Arancione
Violet	Lilla	Blue	Blu
While1	Bianco1	White2	Bianco2

Il modo grafico più semplice da usare è chiamato **bassa risoluzione** (low resolution, o "lores") ed è capace di 40 x 48 pixel. È utilizzabile con Basic Applesoft, ed il suo uso è spiegato nel manuale Introduzione al Basic fornito con Apple IIgs. Non ci sono limitazioni nella collocazione dei colori sullo schermo, e il bit mapping è moderatamente complesso.

La prima pagina occupa lo spazio tra $0400 e $07FF, la seconda va tra $0800 e $0BFF (un Kbyte per ciascuna): è lo stesso spazio occupato dal display del testo, che dunque non può venire usato simultaneamente alla bassa risoluzione.

Nel display di testo ogni byte memorizza un carattere, secondo il codice ASCII: nel display di bassa risoluzione il byte viene considerato diviso in due nibble, ciascuno dei quali conserva le informazioni su un pixel grafico. Se ricordate che un byte conserva un valore tra 0 e 255 noterete che può essere diviso in due informazioni numeriche comprese tra lo 0 ed il 15. Se nel display di testo troviamo 24 righe di 40 colonne allora nella bassa risoluzione trovano posto il doppio di righe e lo stesso numero di colonne, poiché ciascun blocchetto di bassa risoluzione è largo come un carattere ed alto la metà.

Quando ci si è resi conto che sole quaranta colonne di testo non erano sufficienti per sviluppare alcune applicazioni complesse, si è trovato un modo per raddoppiare quella quantità. Il display di testo di 40 x 24 è stato affiancato dal display di 80 x 24.

Dal punto di vista hardware la cosa è piuttosto complessa:

vengono alternati nel display i byte di due banchi distinti di memoria, prima un byte del banco zero e poi un byte del banco uno, poi il secondo byte del banco zero e così via.

Alla raddoppiata capacità dello schermo di testo fa raffronto la raddoppiata capacità del relativo schermo di bassa risoluzione. Con **doppia bassa risoluzione** indichiamo uno schermo di 80 x 48 pixel in sedici colori indipendenti, piuttosto complesso da gestire a causa della collocazione sparsa nella memoria. La doppia bassa risoluzione non è supportata dal firmware: questo significa che non esistono nelle ROM dell'Apple IIgs delle routine già predisposte per usare questo modo grafico, come esistono per la semplice bassa risoluzione.

La doppia bassa risoluzione, pagina uno, occupa la memoria da $0400 a $07FF del banco zero e del banco uno. La seconda pagina occupa lo spazio da $0800 a $0BFF degli stessi banchi: sono due kilobyte per ciascuna.

Nei primi Apple II la massima risoluzione raggiungibile, la **alta risoluzione**, aveva la (per quei tempi) ragguardevolissima capacità di 280 x 192 pixel. Sono utilizzabili solo otto colori, due dei quali duplicati, dunque in pratica sei. I primi quattro colori sono utilizzabili solo nei pixel di posto pari, gli altri quattro solo nei pixel di posto dispari, e dunque la risoluzione orizzontale per quanto riguarda il colore è di soli 140 pixel. Un byte può visualizzare solo il primo od il secondo set di colori, a scelta, ma non un colore di ciascuno. Ogni byte conserva le informazioni di sette pixel contigui sullo schermo.

L'alta risoluzione, pagina uno, si trova tra le locazioni $2000 e $3FFF, la seconda pagina sta tra $4000 e $5FFF. Sono otto kilobyte per ciascuna pagina.

È piuttosto complesso utilizzarla, come avrete intuito dalla seppur breve descrizione precedente, ma con il tempo sono state sviluppate molte tecniche assai ingegnose per utilizzare l'alta risoluzione: di conseguenza quasi tutti i videogiochi disponibili per la famiglia Apple II utilizzano l'alta risoluzione.

In modo analogo a quanto avviene con la bassa e doppia bassa risoluzione, è stato trovato il modo di raddoppiare la capacità orizzontale dell'alta risoluzione, al prezzo di una complicazione ulteriore del bit mapping.

La **doppia alta risoluzione** ha una capacità di 560 x 192 pixel, usa sedici colori (gli stessi della bassa risoluzione) ma diverse prescrizioni vanno rispettate nella loro disposizione.

In memoria si colloca tra $2000 e $3FFF dei primi due banchi di

memoria per la prima pagina, e tra $4000 e $5FFF per la seconda pagina. Sono sedici kilobyte, con due byte per ogni sette pixel.
La estrema complicazione del bit mapping (la stessa Apple l'ha definita in un suo manuale "di una complessità bizantina") ha reso molto arduo programmare routine efficienti che utilizzino la doppia alta risoluzione, e solo di recente si sono visti alcuni videogiochi che la sfruttano, mentre è stata più sfruttata da applicazioni non ludiche (come Instant Pascal) e giochi senza problemi di velocità (le avventure).
La doppia alta risoluzione non è supportata dal firmware di Apple IIgs e dei suoi fratelli minori.
Qualche volta si sente parlare di **media risoluzione**. Si tratta sempre della doppia alta risoluzione, usata con una capacità di 140 x 192 pixel raggruppando i pixel in blocchi di quattro, in modo da consentire il tracciamento di linee colorate continue, rimuovendo le limitazioni sull'accostamento dei colori.

4.2 I nuovi modi grafici dell'Apple IIgs

La nuova modalità grafica introdotta con Apple IIgs prende il nome di **super alta risoluzione** (comincio a chiedermi che nomi inventeranno ancora se con le macchine future si introducessero ulteriori miglioramenti). Nei primi tempi successivi all'introduzione sul mercato del computer, le riviste specializzate ne hanno parlato molto, e quasi tutte a sproposito: vediamo dunque di descriverla da zero.

La super alta risoluzione è uno schermo grafico che può visualizzare sino a 256 colori contemporaneamente, scegliendoli da una palette di 4096 colori. I 256 colori sono divisi in sedici gruppi di sedici, ed ogni riga orizzontale del display sceglie uno dei sedici gruppi come descrittore di propri colori (ogni riga può dunque visualizzare un massimo di sedici colori).
Vi sono 200 righe orizzontali: ciascuna riga può indifferentemente contenere 640 oppure 320 pixel (nulla vieta di mischiare righe con 320 pixel e righe con 640 pixel nella stessa immagine).
Non c'è nessunissima limitazione sull'accostamento dei colori sullo schermo (a parte il buon gusto), e l'immagine (a patto di possedere un buon monitor a colori) è limpida e senza interferenze.

Il bit mapping è il più semplice possibile: la corrispondenza è lineare (cioè il primo byte di memoria corrisponde ai primi pixel e così via sino all'ultimo che corrisponde agli ultimi pixel). Per aiutare il processore a gestire la grafica, esiste una possibilità, chiamata "hardware color fill" che permette di colorare ampie zone dello schermo in velocità: il metodo funziona solo sulle righe da 320 pixel.

Per ogni riga è possibile definire se si desidera generare un interrupt al processore. Questo permette, ad esempio, di modificare le palette mentre il raggio del monitor sta tracciando lo schermo, ottenendo la bellezza di 3.200 colori visualizzabili simultaneamente, e scusate se è poco.

La super alta risoluzione è supportata dal Toolbox di Apple IIgs. Più di cento operazioni, dalle più semplici alle più complesse, dal tracciamento di poligoni colorati alla realizzazione di scritte in ogni stile, sono state sviluppate dai migliori programmatori in Assembler, per ottenere la massima velocità, ed incorporate nella memoria permanente di Apple IIgs, rendendo così possibile ad ogni sviluppatore indipendente di software di creare programmi professionali e veloci.

Nel prossimo capitolo, a beneficio dei programmatori e degli altri interessati, vedremo come viene rappresentata in memoria una immagine grafica ed alcuni principi di programmazione avanzata.

4.3 Il sintetizzatore digitale

I modi grafici dell'Apple IIgs sono molto buoni, ma non i migliori mai visti.

Quel che è semplicemente incredibile in questo computer, invece, di gran lunga superiore al secondo miglior personal computer nel campo, ed anni luce avanti alla corrispondente caratteristica dei modelli precedenti, è l'unità di generazione di suono. La più che autorevole rivista **Byte** ha scritto che il computer andrebbe chiamato "Apple IIgS", con la g di grafica minuscola, ma la s di suono maiuscola.

Phoenix usa il chip **Ensoniq 5503 Digital Oscillator Chip**, abbreviato in DOC. Un componente speciale, chiamato **Sound General Logic Unit** (GLU) svolge i compiti di collegamento tra il

65816 e DOC, alleviando il processore principale dalla maggior parte del peso di gestione.

DOC è in grado di generare 32 voci, tramite altrettanti oscillatori: gli oscillatori sono accoppiati in modo da produrre "solo" quindici voci, ma stereofoniche, mentre un oscillatore è usato per temporizzare l'intero sistema e l'ultimo è lasciato inutilizzato.

DOC produce musica, voce ed effetti sonori secondo lo stesso principio di digitalizzazione usato nei compact disc: il suono viene codificato in forma digitale e conservato nella memoria RAM. Esattamente come nel caso dei compact disc, questo richiede una quantità molto elevata di memoria per sfruttare le capacità del componente: un banco di 64 Kbyte di memoria gli è permanentemente dedicato (è indipendente dalla RAM del 65816). Per sonate di particolare lunghezza e complessità, è possibile far usare a DOC la RAM di Apple IIgs direttamente.

Quando DOC sta dirigendo la sua orchestra di 32 elementi, ciascun oscillatore legge dalla RAM i valori che rappresentano la forma dell'onda sonora da generare, ad una velocità programmabile separatamente per ogni singolo oscillatore. Anche il volume è regolabile per ogni singolo oscillatore.

Quando un oscillatore ha terminato lo spartito, viene generato un interrupt al 65816 che provvede a preparare un nuovo pezzo (mentre durante l'esecuzione è libero di processare indipendentemente dei dati, seppure un po' più lentamente del solito per dover gestire le interruzioni degli oscillatori).

DOC serve gli oscillatori (quando ognuno sta operando contemporaneamente a tutti gli altri) uno alla volta, usando la tecnica del multiplexing. Servono circa 38 microsecondi per servire tutti gli oscillatori, cioè essi vengono controllati 26320 volte al secondo.

Sul retro di Apple IIgs, sulla estrema destra, si trova una presa minijack stereo per collegare il generatore di suono ad un complesso stereofonico esterno – normalmente, cioè quando nulla è collegato a quella presa, il suono viene riprodotto attraverso il piccolo altoparlante monofonico incorporato nel computer.

Sulla piastra madre di Apple IIgs si trova un connettore molex a sette pin (alla locazione J 25): si può prelevare il segnale di output del generatore di suono e processarlo ulteriormente. Lo **hardware manual** lo descrive nei dettagli tecnici e spiega come usarlo per ottenere sino a 15 canali sonori indipendenti. Anche

più interessante è la possibilità di usare quello stesso connettore per trasmettere del suono in input a DOC, in modo da digitalizzarlo e memorizzarlo. Non passerà mòlto prima che ci ritroviamo ad ascoltare Von Karajan dalla viva voce di Phoenix.

Il generatore di suoni è supportato dal firmware di Apple IIgs: il toolbox (vedi capitolo 5 e capitolo 9) mette a disposizione due tool composti da numerose chiamate di sistema anche di alto livello per consentire un uso di DOC semplificato ma sofisticato.

5

PROGRAMMAZIONE DI GRAFICA E SUONO

Una panoramica a volo d'uccello sui metodi vecchi e nuovi per elaborare programmi che fanno uso della grafica.

5.1 La super alta risoluzione in memoria

Lo schermo di super alta risoluzione occupa normalmente lo spazio di memoria RAM da $E1/2000 a $E1/9FFF. Di questo spazio, 32 Kbyte in tutto, gran parte è occupato dall'immagine bit mapped, mentre un piccolo spazio viene riservato alla palette ed ai descrittori di riga: tutti termini che spiegheremo tra poco. Quando una immagine viene salvata su disco, vengono salvati anche i descrittori e la palette (la tavolozza dei colori usati nell'immagine), creando così un file di 65 blocchi (64 blocchi di informazioni, cioè 32 Kbyte, e un blocco per il sistema operativo). Il file deve per convenzione essere di tipo BIN oppure FOT.

La super alta risoluzione disposta con ogni riga in modo 640 pixel di larghezza è composta di 128.000 distinti pixel. È praticamente il massimo visualizzabile da un normale monitor ad alta risoluzione; quando ai progettisti di Apple IIgs è stato chiesto perché non avessero incluso un modo grafico da 640 x 400 pixel, così come è stato fatto nella Amiga di Commodore, essi hanno fatto notare che quella capacità non viene mai usata da nessun

programma proprio perché nessuno dispone dei costosissimi monitor speciali.

La memoria per l'immagine bit mapped è quella compresa tra $E1/2000 e $E1/9CFF, e come abbiamo detto è disposta in modo continuo. Se tutte le righe sono composte di 320 pixel, il primo byte ($2000) contiene la rappresentazione dei primi due pixel dello schermo, quelli nell'angolo in alto a sinistra. Il byte $2001 corrisponde al terzo e quarto byte della prima riga e così via, sino al byte $209F che corrisponde ai due pixel più a destra della prima riga. Il byte successivo, il $20A0, è legato ai primi due pixel di sinistra della seconda riga. Si procede sino al byte $9CFF che corrisponde ai due pixel in basso a destra nello schermo.

Dato che ogni byte è composto da otto bit, ogni pixel ha a disposizione 4 bit (quantità che qualcuno chiama un **nibble**) e può quindi assumere un colore a scelta tra sedici. Quali colori siano i sedici in questione viene scelto sistemando la palette, come stiamo per scoprire.

Ogni riga, indifferentemente composta da 320 o da 640 pixel, è rappresentata da 160 byte (in esadecimale, $A0 byte).

Quando una riga è disposta in modo da essere composta da 640 pixel, ogni byte corrisponde a quattro, e non due, pixel. Ogni pixel ha a disposizione due bit, e può dunque assumere solo uno tra quattro colori scelti nella palette.

Nella memoria tra $E1/9D00 e $E1/9DFF vanno i descrittori delle righe, che la documentazione tecnica Apple chiama "puntatori": preferisco evitare quel termine, dato che questi puntatori non hanno nessuna parentela con i puntatori dei linguaggi di programmazione come C o Pascal.

A ciascuna riga del display è lasciato un byte in questo spazio: per la prima riga il byte $9D00, per la seconda il byte $9D01 e così via sino all'ultima e duecentesima che usa il byte $9DC7 (i byte da $9DC8 a $9DFF non sono usati e vanno mantenuti azzerati).

Il significato di questi byte determina le caratteristiche di ciascuna riga: ogni bit nel byte ha un differente significato.

Il bit meno significativo, il bit zero, determina se la riga deve avere 320 oppure 640 colonne (pixel). Se il bit è azzerato, la riga avrà 320 pixel.

Il bit uno può venire usato per sincronizzare un programma alla scansione del pennello elettronico del monitor che traccia l'immagine. Se il bit è settato, se vale 1, verrà generato un interrupt al processore 65816 quando il pennello sta iniziando a trac-

ciare la riga. Questo può permettere l'installazione di una routine che gestisce il cambio di palette per ogni passata del video, la gestione di animazioni senza sfarfallio ed altre mirabilie.

Il bit numero due determina l'uso dello Hardware Color Fill Mode. Si tratta di una modalità disponibile solo quando la riga è disposta per 320 pixel (in caso contrario il valore del bit è ignorato), ed è particolarmente utile per manipolare in fretta i colori. Immaginiamo di voler tracciare una riga interamente rossa sullo schermo. Sia il rosso il colore numero 5 della palette (vedremo tra poco come scegliere i colori della palette): normalmente dovremmo riempire i 160 byte corrispondenti di pixel della riga con il valore $55. Questo richiede un certo tempo.

Se il bit del Color Fill è settato (vale 1), è sufficiente mettere nel primo byte della riga il valore $50 e lasciare azzerati gli altri byte della riga: i circuiti dei computer genereranno quel colore su tutta la riga. Se vogliamo che la riga sia per la prima metà rossa e per la seconda metà azzurra, associamo all'azzurro il valore (ad esempio) 8, e mettiamo il valore $80 nel byte di mezzo (l'ottantesimo byte visto che una riga conta 160 byte).

Il bit numero tre nel byte non è usato, e va mantenuto azzerato.

I quattro bit più significativi, i bit numero 4-7, indicano una delle sedici palette. In questo modo la riga viene accoppiata ad uno degli insiemi di sedici colori. Esistono sedici di questi insiemi per un totale di 256 distinti colori al massimo (fatti salvi trucchi leciti e meno).

La zona per definire le palette è quella che va dall'indirizzo $E1/9E00 all'indirizzo $E1/9FFF. In questi 512 byte vanno definite le sedici palette ciascuna composta da sedici colori. Ogni palette ha dunque a disposizione 32 byte di spazio: lo spazio per la palette zero è rappresentato dai byte tra $9E00 e $9E1F, quello per la seconda palette sta tra $9120 e $913F e così via sino all'ultima che usa i byte da $9FE0 a $9FFF. Non c'è spazio inutilizzato come nella zona dei descrittori.

Nei 32 byte di ogni palette vanno descritti sedici colori: due byte per ogni colore. Dato che i colori nell'Apple IIgs sono "soltanto" 4.096 e dato che una word (due byte) può memorizzare un numero intero compreso tra lo zero e il 65.535 c'è spazio d'avanzo.

I colori usati nell'immagine sono scelti come abbiamo accennato nel paragrafo 3.1: ognuno dei tre colori fondamentali, ros-

so verde e blu, può assumere una forza tra lo zero ed il quindici, e la combinazione dei tre colori fondamentali genera i colori della scala cromatica. Le combinazioni possibili sono (libro di matematica alla mano) 16 elevato alla terza potenza ovvero 4.096.

Il primo colore della prima palette viene definito nei byte $9E00 e $9E01. Nel primo byte (quello pari) vengono scelte le intensità del verde e del blu: i quattro bit più significativi corrispondono al verde; nel secondo byte (quello dispari) i quattro bit più significativi debbono essere mantenuto azzerati, ed i quattro meno significativi individuano l'intensità del rosso.

Ripetendo la cosa per 256 volte si sono definiti tutti i colori utilizzabili.

Notate che non necessariamente i 256 colori debbono essere tutti differenti tra di loro: GS Paint, ad esempio, usa soltanto 16 colori per i disegni più qualche altro per la mela nell'angolo in alto a sinistra. Il sistema più spiccio per usare solo sedici colori consiste nel creare la propria palette iniziale e poi ricopiarla nello spazio delle altre quindici palette distinte.

Se in tutti i 256 spazi di due byte viene messo lo stesso colore, lo schermo si presenta monocromatico, anche se nello spazio dell'immagine c'è qualcosa di significativo.

Un'ultima cosa prima di concludere la trattazione della connessione memoria-immagine: esiste la possibilità di effettuare lo shadowing della super alta risoluzione nel banco 01 di memoria, in modo analogo a quanto viene fatto per tutte le altre modalità di display quando Phoenix sta emulando i vecchi Apple II. Si tratta di una possibilià solo moderatamente interessante, che cito per completezza.

5.2 La compattazione

Come abbiamo già accennato, quando l'immagine viene salvata su disco tutte e tre le zone di memoria attigue che la definiscono (bit image, descrittori e palette) sono slvate: questo crea un file di 65 blocchi su disco, ovvero 32 Kbyte e mezzo. Come accade sempre, una ottima risoluzione consuma grandi quantità di memoria: anche disponendo di un disco microfloppy da tre pollici e mezzo (vedi capitoli 6 e 7 per una introduzione al DOS e ai floppy) con una capacità di 800 Kbyte, solo 23 immagini pos-

sono trovarvi posto. Questo vale anche se le immagini sono molto semplici.

Sino dai primi modelli di Apple II la compattazione delle immagini di alta risoluzione è stata una tecnica ampiamente in uso: introdotta per consentire la creazione di avventure grafiche di una certa complessità è poi diventata di uso comune anche nei videogame più complessi. L'algoritmo più comunemente usato per la codifica e decodifica è molto semplice: iniziando dal primo pixel in alto a sinistra nello schermo, si scrive un valore che identifica il colore del pixel, seguito da un byte che specifica quanti pixel identici seguono. Se l'immagine è particolarmente semplice può venire realizzato un risparmio di spazio in memoria e/o su disco sino al 94% circa.

Oggi la tecnica di compattazione è ancora più conveniente del solito: infatti esiste nel Toolbox una coppia di routine addette proprio a questo scopo. Questo significa che esiste uno standard in proposito, ovvero che una immagine compattata da un programma può venire facilmente usata da un altro, senza dover usare come tramite la forma decompattata.

Le immagini di super alta risoluzione compattate con le routine di QuickDraw II devono venire salvate su disco nei file di tipo $C0.

5.3 L'animazione e le istruzioni move

Che venga usata in un videogioco o in un programma didattico, l'animazione delle figure di alta risoluzione è da sempre un tema studiato dai programmatori, dagli appassionati del Basic ai professionisti dell'Assembler.

Ed è tutt'altro che semplice da realizzare. Il metodo più semplice per animare, dunque muovere, una figura sullo schermo è di spostare i byte che ne costituiscono la bit image nello spazio del display.

Il guaio di questo metodo è che se la velocità non è più che ragguardevole, l'occhio, ricettore molto fedele, percepisce uno sfarfallio nell'immagine, riconoscendo l'intervallo di tempo che separa il momento in cui la figura viene cancellata dalla vecchia posizione ed il momento in cui viene disegnata nella nuova posizione.

Nei computer che dispongono di un chip schiavo per la grafi-

ca, viene spesso usato il costrutto chiamato **sprite:** la figura viene mossa dallo hardware, e non dal programma di controllo, raggiungendo così una velocità più che sufficiente ad evitare lo sfarfallio. Vi sono limitazioni nella quantità di sprite che possono essere simultaneamente presenti sullo schermo e nelle loro dimensioni, ma solitamente l'uso degli sprite risolve il problema, eccetto che per scopi speciali.

Pur non possedendo attrezzature hardware per la generazione degli sprite, l'Apple II ha sempre sfoggiato animazioni più che degne: merito della tecnica conosciuta come **page flipping**, ovvero scambio delle pagine. Il principio è questo: dato che esistono due distinte zone della memoria visualizzabili come schermi di alta risoluzione, il disegno avviene su una delle due pagine, quella nascosta, mentre la pagina visualizzata sul monitor non viene toccata. Quando la pagina nascosta è stata preparata con il "fotogramma" successivo viene messa in vista, e la pagina che era visibile sparisce: il ciclo può ora ripetersi cancellando e ridisegnando la pagina nascosta.

Su questa tecnica di base si sono poi andate sviluppando numerose varianti e raffinatezza, come l'uso della cosiddetta "pagina tre" per lo sfondo, le shape ed altro ancora. È un argomento piuttosto interessante sul quale mi riprometto di tornare in futuro, in un libro successivo, se ne avrò l'occasione.

Il problema è che la super alta risoluzione ha solo una pagina, come abbiamo visto nell'ultimo capitolo, a differenza di tutti gli altri modo di display. Ne consegue che le animazioni vanno fatte in velocità. Fortunatamente, tra le aggiunte e le migliorie che distinguono il processore 65816 dal suo predecessore 65C02 vi sono due istruzioni, MVP e MVN, per muovere blocchi consecutivi di byte nella memoria (rispettivamente in positivo ed in negativo, cioè dal primo all'ultimo e dall'ultimo al primo). Disgraziatamente, le due istruzioni non sono granché efficienti, e data anche la grande risoluzione dello schermo, non sono in grado di trasferire i 32 kilobyte che costituiscono una immagine di super alta risoluzione in un tempo sufficiente ad evitare lo sfarfallio.

Tanto per saperlo, il pennello che ridisegna lo schermo compie la sua operazione in un cinquantesimo di secondo: le istruzioni di block move richiedono circa cinque volte più tempo per spostare i 32 Kbyte. Il risultato è evidente quando fare eseguire una animazione a GS Paint (purché l'animazione non sia limitata ad una piccolissima parte dello schermo, come nel caso della ra-

gazza che strizza l'occhio): ci sono cinque o sei punti sullo schermo dove l'immagine traballa.

La soluzione, a mio parere, è di usare gli interrupt di linea per sincronizzare il movimento di parti dello schermo con il pennello: abbiamo visto come sia possibile stabilire linea per linea se debba essere generato un segnale di interruzione al microprocessore. Una routine sufficientemente intelligente da innescare quei segnali per il minor numero di linee possibile (in modo da minimizzare l'overhead) e da usare il segnale per muovere i byte nella parte dello schermo appena tracciata dovrebbe poter evitare lo sfarfallio: resta il fatto che ci vorranno cinque passate del pennello prima di avere l'immagine postdatata, e dunque si vedrà per qualche frazione di secondo l'immagine un po' modificata ed un po' no. Anche questo può essere trasformato da uno svantaggio in un vantaggio usandolo come elemento dell'animazione, ma bisogna pensarci un po' su.

Dato che il colore dei pixel è controllato innanzitutto dai valori delle palette, che occupano soltanto due pagine (512 byte) di memoria, le animazioni di colore sono molto semplici da eseguire, e potenzialmente molto interessanti. Per esempio è semplice eseguire il *fade in*, cioè l'effetto che vede l'immagine apparire lentamente dallo schermo nero, schiarendosi piano piano sino ad arrivare alla luminosità corretta.

5.4 Grafica, suono e la routine wait

La doppia velocità di Apple IIgs introduce molti benefici e qualche perplessità per l'utente. Spesso questi si vede costretto ad un continuo entra ed esci dal Pannello di Controllo per sistemare le cose: rallenta i videogiochi, accelera gli applicativi...

Apple IIgs opera ad 1,024 megahertz di velocità quando predisposto sull'opzione "lento". Quando invece viene fatto lavorare a piena velocità i megahertz aumentano a 2,8: questa velocità è mantenuta nell'eseguire le routine della ROM, programmi memorizzati in quest'ultima, come avviene in tutti i computer: in nanosecondi, e sono mascherati a FPI quando il processore sta lavorando sulla ROM. La diminuzione della velocità per programmi in RAM, invisibile, è esattamente dell'otto per cento, rallentando il 65816 a circa 2,57 MhZ.

Per il programmatore la velocità è un problema relativo: chiamando il toolbox è possibile scegliere la velocità preferita per ogni singola routine. Per qualche particolare la doppia velocità può dare dei fastidi: per esempio, programmando un applicativo con la velocità maggiore non si possono usare le routine sonore sviluppate in passato per dare indicazioni acustiche senza modificarle, perché la velocità tripla deforma irriconoscibilmente i suoni: e usare il DOC per fare un semplice buzz è come usare una bomba atomica per schiacciare una zanzara. È interessante notare che la buona e vecchia routine WAIT di System Monitor è stata modificata per tenere conto della differenza di velocità, e funziona correttamente a prescindere da questa.

WAIT si trova all'indirizzo $FCA8: vi si entra con un valore nell'accumulatore che indica la durata del tempo da lasciare passare, ed il controllo vi viene restituito dopo quel tempo esatto.
Bisogna chiamarla in modo emulazione, come per tutte le routine di Monitor; i registri X ed Y sono restituiti senza modifiche, e l'accumulatore è azzerato.
La formula per calcolare il ritardo in microsecondi effettuato da WAIT in funzione del valore contenuto nell'accumulatore è:

$$\frac{1}{2} (26 + 27A + 5A^2) * 14 / 14{,}31818$$

L'ultimo rapporto (14/14,...) è la correzione per tener conto della differenza tra la velocità reale del computer e la velocità teorica di un megahertz, ed è tale da poter essere trascurato in quasi tutti i casi.

6

IL DISK JOCKEY

Cosa sarebbe del computer senza i suoi dischi magnetici? In questo capitolo parliamo dei disk drive (o lettori di dischi, se preferite la dizione italiana) per l'Apple IIgs e del modo in cui rendono possibile salvare i programmi.

Non ci interesseremo della natura fisica di questi aggeggi, se non marginalmente: come nel resto del libro, il capitolo spiega le ingegnose trovate usate nell'Apple II partendo da qualche dato di fatto accettato.

Esistono svariati modelli di disk drive per Apple II. Invece di limitarci a citare i tipi creati specificamente per Phoenix, li passeremo brevemente in rivista tutti: questo dovrebbe consentirvi di usare con il vostro computer anche unità disco che possedete già e che venivano usate con un modello precedente.

Un disco, in sé, è solo un aggeggio piatto coperto da 0,8 millimetri di polvere magnetica: per poterlo usare per salvare i programmi, dobbiamo fissare delle convenzioni sul trattamento fisico da riservargli. Non solo: dobbiamo fornire al computer un programma in ROM che permetta l'accesso al disco, che governi insomma l'unità hardware del disco (disk drive); quest'ultimo va sotto il nome di DOS, o sistema operativo del disco (Disk Operating System). Nel capitolo sette parleremo in dettaglio dei sistemi operativi dell'Apple IIgs.

6.1 Il Disk II

Quando l'Apple II venne creato, tutto il salvataggio dei dati su supporto magnetico veniva effettuato esclusivamente grazie alle normali cassette, con un registratore collegato al computer tramite una presa DB5 sul retro, così come fanno ancora oggi alcuni semplici home computer.

Le cassette, però, sono spaventosamente lente e non garantiscono sufficiente sicurezza. Così, gradatamente il sistema delle cassette venne sostituito da quelli dei dischi magnetici quando casa Apple introdusse un disk drive per gli Apple II nel 1978. Si chiamava Disk II ed era prodotto nella meccanica interna di precisione dalla Shugart.

Detto per inciso, le cassette restarono disponibili per gli inguaribili passatisti, per gli appassionati in ristrettezze economiche (il sottoscritto ha usato il suo primo Apple II, modello II+, per qualche mese unicamente con cassette, prima di concedersi il lusso di una unità a disco Disk II) e per i pirati del software (in quanto la scrittura di programmi su disco può essere impedita, ma non si poteva inibire la cassetta).

Co l'Apple IIc si decise di eliminare quella che era ormai un'obsolescenza inutile, e vennero rimossi i supporti hardware e software di base per l'uso delle cassette, che dunque non esistono neppure nel successivo Apple I Igs.

Il Disk II usa dischi cosiddetti da cinque pollici di un quarto. È questa la misura del loro diametro, in centimetri tredici e mezzo; iste anche una misura maggiore, da otto pollici, che sta cadendo in disuso, ed una inferiore da tre pollici e mezzo che però non esisteva ancora nel '78. Un sinonimo dei nomi dei dischi in pollici è *floppy disk* per gli otto pollici, *minifloppy disk* per i cinque pollici e un quarto, *microfloppy disk* per i tre pollici e mezzo. Floppy significa flessibile ed indica la capacità del supporto di essere piegato leggermente senza riportarne danno (i microfloppy soono conservati in custodie di plastica rigida e quindi è fisicamente impossibile piegarli senza infrangere il pezzo).

I dischi Apple da cinque pollici e un quarto sono usati a singola faccia. Questo significa che esiste nel drive una sola testina di lettura e scrittura, che agisce dunque su un solo lato del disco. Normalmente undisco da cinque pollici e un quarto – che trovate scritto come 5'' 1/4 – viene impiegato su due lati estranedolo dal drive, capovolgendolo e reinserendolo di modo che la testi-

na acreda al secondo lato. La maggior parte dei computer creati più di recente, valga come esempio per tutti il PC della IBM, usano contemporaneamente le due facciate: i loro drive sono detti a *doppia faccia*. Ovviamente il drive costa di più, perché deve essere dotato di una doppia testina.

I dischi usati da Disk II e dai suoi successori sono inoltre a *singola densità*, e con trentacinque *tracce* per facciata.

Dire che i dischi di Apple II – come quelli del Commodore 64, per esempio – hanno 35 "tracce", significa dire che sono divisi in 35 solchi circolari e concentrici; sino al 1980 ciascuna traccia era ulteriormente suddivisa in 13 "settori". Un settore registra 256 byte di informazioni (cioè 4 settori conservano un kilobyte).

Nel 1980 si migliorò lo hardware dell'interfaccia disco permettendo l'introduzione di 16 settori per traccia, portando così la capacità del disco da 115 kilobyte per facciata a 140.

Un disco 5"1/4 può essere usato – se il drive è più sofisticato e costoso dei Disk II e se la qualità del disco lo permette – a doppia densità, raddoppiando il numero delle tracce. Questo significa che le tracce sono del doppio più fitte: la cosa è possibile perhcé la testina magnetica ha una risoluzione migliore.

Visto che siamo sull'argomento della capacità dei dischi, voglio attirare la vostra attenzione ad un punto piuttosto controverso. Troverete qualche volta scritto che i dischi Apple contengono 143 Kbyte di informazioni, e qualche altra volta vedrete scritto 140 K.

Nessuna contraddizione in questo: tutto sta nello stabilire cosa sia un kilobyte.

Scusatemi ora se spiego qualcosa che per la maggior parte dei mie lettori sarà banale per arrivare al punto. La unità più piccola di informazione che sia possibile è chiamata **bit,** e corrisponde a una risposta Si/No. Unendo otto bit abbiamo un **byte,** una unità di misura a sua volta. In un byte facciamo stare un numero intero compreso tra zero e duecentocinquantacinque, oppure un carattere dell'alfabeto, o una informazione di quell'ordine di grandezza: la memoria dei computer si misura in byte e nei loro multipli (se il discorso non vi quadra o se volete saperne di più procuratevi un libro introduttivo ai computer in generale).

Ed eccoci al punto dolente. Secondo lo standard scientifico, i multipli di una unità di misura vanno di mille in mille ed usano prefissi comuni: per esempio ci sono mille metri in un kilometro, mille litri in un kilolitro, mille watt in un kilowatt e mille kilowatt in un megawatt.

Con i byte non è così. In un kilobyte ci sono 1.024, e non mille, byte. Questo dipende dal modo in cui il computer opera internamente: risulta molto più comodo usare il sistema binario (le potenze del due) piuttosto che il sistema decimale (le potenze del dieci). Milleventiquattro è una potenza del due ma non del dieci.

Un disco Apple II ha dunque la capacità di 140 Kbyte, cioè di 143.360 byte. Qualche volta qualcuno si ente obbligato a scrivere 143 k: niente di male, si stanno solo cambiando le unità di misura, finché le idee restano chiare.

Torniamo al II. È solo dall'avvento dell'Apple IIc che le componenti elettroniche che consentono l'allacciamento dell'unità disk drive dal computer sono inglobate nel computer. I modelli precedenti debbono inserire in uno slot una apposita interfaccia e collegare il disco a questa interfaccia.

Di conseguenza, è possibile usare un Disk II su un Apple IIgs solo inserendo la sua interfaccia in uno slot (normalmente il 6) e quindi abilitando quell'interfaccia servendosi del Pannello di Controllo.

Non è possibile attaccare altri drive ad un Disk II nella maniera detta "a cascata": una interfaccia permette di collegare a se un massimo di due Disk II.

6.2 Il Drive Esterno IIc e il DuoDisk

Quando il IIc venne creato nel 1984 Apple stabilì di rendere standard la dotazione di un disk drive 5" 1/4 – che nel IIc è incorporato. Il Drive esterno IIc viene collegato all'Apple IIc per disporre di una seconda unità disco oltre a quella incorporata. Il IIc può aggiungere altri dischi esattamente nello stesso modo usato da Apple IIgs: in cascata.

Questo significa che i drive sono collegati l'uno all'altro in catena, con il primo della serie collegato al computer attraverso la porta disco. Nei precedenti modelli di Apple II i drive venivano collegati in parallelo, cioè ciascuno di essi doveva venire collegato con un cavo direttamente al computer attraverso una interfaccia.

Il Drive esterno IIc è stato creato per soddisfare queste scelte. A differenza dei precedenti modelli, è un drive cosiddetto *slim*

line, cioè sottile, alto la metà del precedente Disk II. Può venire collegato direttamente all'Apple IIgs attraverso la porta dischi oppure collegato ad una catena di dischi.

Deve però essere l'ultimo disco della catena, poiché non dispone di una presa per collegarvi altri drive: questa scelta è dovuta al fatto che lo Apple IIc, un computer portatile, non può soddisfare la richiesta di energia elettrica di più di un disco 5" 1/4 esterno. Rendendo fisicamente impossibile il collegamento in catena di più di un disco si evita la possibilità di sforzare l'alimentatore del IIc.

A parte questa limitazione, il Drive esterno IIc (che viene qualche volta chiamato nella documentazione tecnica Apple con il nome di UniDisk 5" 1/4, ma non va confuso con lo UniDisk 3" 1/2 di cui parleremo nel prossimo paragrafo) è perfettamente adatto ad un Apple IIgs.

Per soddisfare le necessità del modello precedente, l'Apple IIe, è stato creato il DuoDisk, che riunisce in un solo componente due disk drive 5" 1/4. Come il Drive esterno IIc, anche il DuoDisk può essere unito in cascata a Phoenix, purché sia l'ultimo elemento della catena in quanto non dispone della presa per collegamente a sé di altri elementi. Non possono essere collegati in cascata i più vecchi DuoDisk, quelli aventi un numero seriale inferiore a 433.754.

6.3 L'UniDisk

In tempi recenti la tecnologia ha reso pian piano superati i dishi Apple da soli 140 Kbyte per facciata. Apple Computers si è trovata di fronte ad una necessità: creare una unità a disco più capace, se possibile più veloce e più "intelligente" (il che significa che richiede meno attenzioni da parte del microprocessore, il cervello del computer, e programmi più semplici per controllarla).

Probabilmente la scelta più semplice sarebbe stata la creazione di una unità 5" 1/4 a doppia densità, magari a doppia faccia, che avrebbe portato la capacità di un singolo dischetto a 560 Kbyte.

Notate che stiamo parlando di 560 Kbyte contro i soli 360 per un disco di PC IBM dalle stesse caratteristiche: questa notevole

differenza dipende dalla incredibile potenza di un circuito integrato chiamato IWM che viene montato sugli Apple II. La sigla significa Integrated Wozniak Machine: macchina integrata di Wozniak, un'altra delle invenzioni rivoluzionarie dell'inventore dell'Apple II. IWM viene usata anche su Apple Macintosh.

Apple ha fatto una scelta diversa. Sin dall'introduzione dello sfortunato Lisa, Apple aveva scelto la nuova tecnologia dei drive da tre pollici e mezzo (che trovate scritto anche come 3" 1/2), usati poi anche nel Macintosh come doppia densità e singola faccia per 400 Kbyte di capacità.

Per l'Apple II è stato sviluppato un modello migliorato del disk drive di Macintosh: a doppia faccia, di modo che la capacità divenisse di ben 800 Kbyte (circa un anno dopo, adottato anche da Macintosh). A differenza del modello di MacIntosh, il disco da 3" 1/2 per Apple II è intelligente: contiene infatti un processore 65C02, il fratello minore di 65816, interamente dedicato a gestire l'utilizzo del disco.

UniDisk è il nome di questo disco. Può essere collegato in cascata, può essere attaccato direttamente alla porta dischi di Apple IIgs come a quella dell'Apple IIc per il quale è stato creato.

6.4 L'Apple 3.5 Drive e l'Apple 5.25 Drive

Arriviamo ora ai disk drive creati esplicitamente per Phoenix. Sono dello stesso color grigio platino dell'Apple IIgs, e sono stati chiamati Drive 3.5 e Drive 5.25 senza molta fantasia, dalle dimensioni dei dischi.

Ovviamente, nessuno dei due presenta problemi nel collegamento con il IIgs, e vengono collegati attraverso la porta dischi.

L'Apple 5.25 drive è il fratello gemello del Disk IIc esterno: l'unica differenza è la presenza sul retro di una presa per collegare in cascata altri drive. A causa della presenza dell'elettronica di controllo di questa porta, è un poco più alto del drive per IIc.

L'Apple 3.5 drive è invece una versione modificata dell'UniDisk. La differenza più notevole è la capacità del Disk 3.5 di essere usato indifferentemente da un Apple II o da un Apple Macintosh. Se possedete entrambi i computer siete in grado di usare un solo drive per entrambi, realizzando un notevole risparmio economico. Nonostante la "perdita di intelligenza" rispetto all'UniDisk (Apple 3.5 Drive non contiene infatti il processore 65C02) il disco è veloce quanto il predecessore.

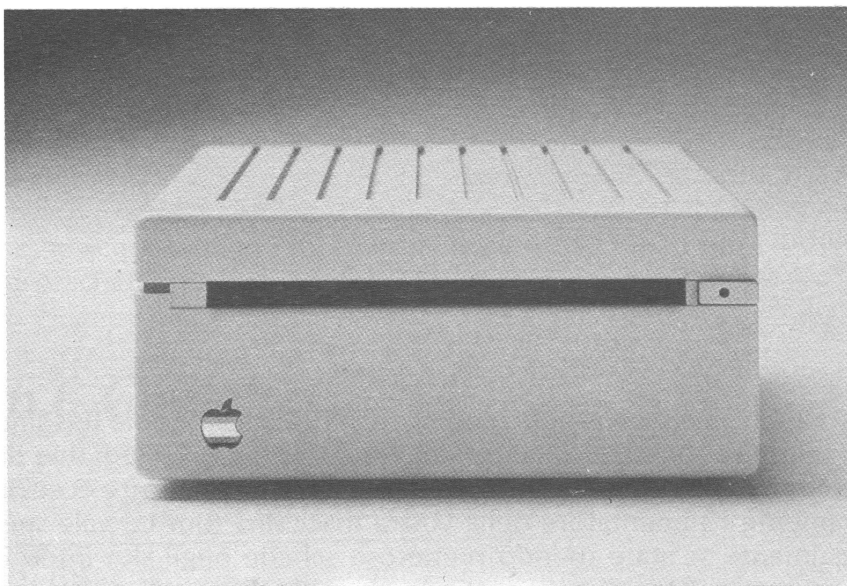



Fig. 1. Apple 3.5 drive.

C'è un problema che vi affliggerà se pensate di usare contemporaneamente due dischi da 3" 1/2 con il vecchio sistema operativo.

Il problema si manifesta usando una delle prime versioni di ProDOS, la 1.0 o la 1.1, oppure il Pascal 1.3: come colpito da una maledizione, il secondo disco da 3 pollici svanisce dalla lista dei dischi disponibili. Pur essendo fisicamente presente, non viene visto dal sistema operativo – al suo posto appare invece il disco RAM selezionato con Pannello di Controllo. Il problema non sussiste invece se utilizzate una delle versioni più recenti del sistema operativo, ragion per cui il problema è di scarsa rilevanza.

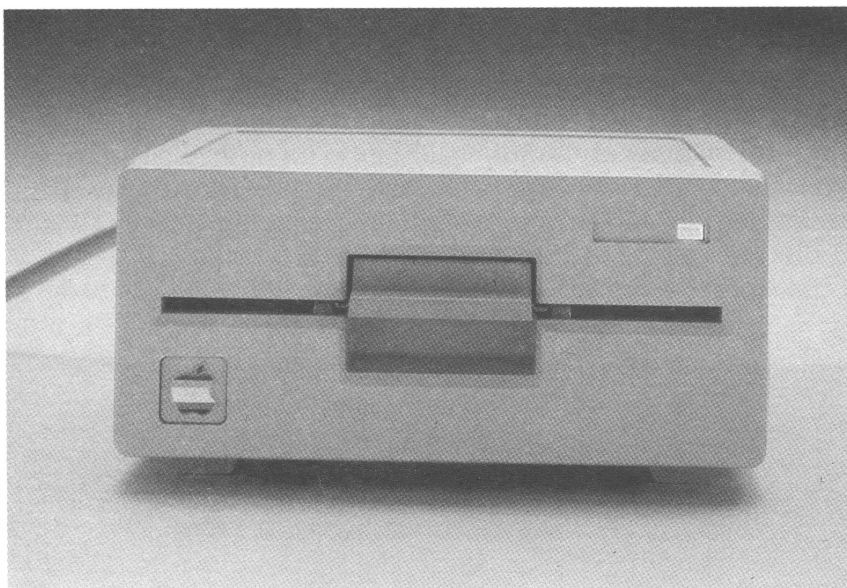



Fig. 2. Apple 5.25 drive.

Notate che è consentito collegare in cascata all'Apple IIgs sino a sei unità a disco di qualunque dimensione (non più di due da cinque pollici ed un quarto). Tuttavia, per non affaticare eccessivamente l'alimentatore della vostra macchina (questo vale specialmente se state usando numerose schede negli slot interni) dovreste evitare di collegare alla porta dischi più di quattro unità, due da 3" 1/2 e due da 5" 1/4. Non c'è certo da lamentarsi, dato che si tratta di quasi due megabyte in linea.

Il disco RAM e quello ROM non contano nel fare il calcolo dei drive collegati.

6.5 Il ProFile

Profile è il nome del disco rigido della Apple Computer. Un disco rigido *(hard disk)* è un disco di grandi dimensioni e gran-

dissima capacità che resta permenentemente nel disk drive (viene detto in gergo **non rimovibile**), sotto vuoto spinto per consentirne un funzionamento impeccabile.

I primi ProFile della Apple avevano una capacità di 5 Megabyte (5.120 Kbyte), i modelli più recenti hanno capacità doppia. Lo Apple Hard Disk 20 è un ProFile da 20 megabyte di capacità creato appositamente per Apple IIgs.

Il ProFile, che è superiore a tutti i dischi che abbiamo già trattato per velocità di risposta e per fedeltà, non viene collegato attraverso la porta dischi dell'Apple IIgs, ma richiede una interfaccia apposita in uno slot. Se si lavora sempre in ProDOS e se non si usa la rete di comunicazione AppleTalk è una buona idea installare la scheda nello slot sette: in queto modo Apple IIgs effettuerà lo startup da ProFile.

Attenzione: se possedevate già un ProFile con relativa interfaccia dovete far eseguire una piccola modifica a questa, che non può funzionare perfettamente in un Apple IIgs. Ovviamente, se acquistate un ProFile insieme all'Apple IIgs vi verrà fornita una scheda già aggiornata.

6.6 Come scegliere una configurazione adatta alle proprie necessità

Nei primi cinque paragrafi di questo capitolo abbiamo parlato delle unità disco collegabili all'Apple IIgs per risolvere i problemi di quanti vogliono continuare ad usare le vecchie unità con il nuovo computer.

E venuto il momento di fare un discorso un poco più ampio. Quali e quanti dischi servono davvero? In questa breve discussione vedremo di considerare i pro e i contro.

Per alcuni lettori la scelta delle unità disco può essere vincolata soprattutto dalle disponibilità economiche. Studenti, hobbisti che hanno voluto cambiare un vecchio Apple II con il nuovo IIgs, potrebbero essersi trovati di fronte a problemi di disponibilità e in dubbio.

La prima considerazione va fatta osservando il nuovo sistema operativo ProDOS 16, creato per sfruttare al meglio Phoenix:

ProDOS 16, comprendendo in questo termine tutti i documenti connessi e che vanno messi su un unico disco, richiede circa 300 Kbyte di spazio su un disco.

Questo ci porta subito ad una conclusione: per poter usare Apple IIgs è necessario possedere almeno un disk drive da 800 Kbyte di capacità, uno UniDisk o un Apple 3.5 Drive (preferibilmente il secondo).

Certo, è possibile limitarsi all'uso dei dischi 5" 1/4. Il computer funziona perfettamente anche così: ma in questo modo si possono usare solo programmi basati su ProDOS 8 o su un altro sistema operativo, cioè solo programmi creati per tutti gli Apple II, cioè si finisce per usare Apple IIgs come un modello precedente. E a questa stregua tanto vale risparmiare qualche lira e comprarsi un Apple IIc.

Un disco da 5" 1/4 vi servirà con quasi altrettanta certezza. Il punto forte da sempre dell'Apple II è il suo straordinario patrimonio di software, la quantità incredibile di programmi che sono stati scritti per la piccola grande mela. Una percentuale notevole di questi programmi è stata creata su dischi da cinque pollici, spesso in modo tale che non ne è possibile il travaso sui più capaci dischi da tre pollici.

Dunque, un disco da 5" 1/4 è una necessità quasi assoluta. A meno che, per qualche motivo, il vostro Apple IIgs venga usato solo con un certo programma. Per esempio, con AppleWorks, oppure per usare esclusivamente le sue capacità grafiche in un impiego artistico o creativo. Potete fare a meno di un disco 5" 1/4 se usate Apple IIgs con un numero ridotto di programmi, basati su ProDOS che possedete in versione su microdisco.

Serve un secondo disco da 800 Kbyte? Si e no.

Se avete grandi necessità di spazio su disco, un secondo disco 3.5 è consigliabile: oppure potreste indirizzarvi all'acquisto di un disco rigido.

Un secondo disco da 3" 1/2 diventa indispensabile se avete necessità di creare spesso copie dei vostri dischi: in quel caso il procedimento di copia diventa semplicissimo con due dischi, ed infernale con uno solo, perché si trasforma in una serie di "inserisci il disco originale" "inserisci il disco copia" "inserisci il disco originale" eccetera. Un secondo disco da 3.5 vi serve assolutamente se per qualche motivo deciderete di non aver bisogno di nessun drive da 5" 1/4. Se siete dei programmatori che utilizzano lo Apple Programmer's Workshop troverete indispensabili due drive da 800 Kbyte ed un megabyte di memoria centrale: un

drive deve contenere il disco di sistema con il sistema di sviluppo, i tools e le librerie, il secondo servirà per il programma sviluppando.

Avendo la possibilità dell'acquisto di un secondo disco questa non va scartata. Ma non si tratta di una necessità stringente, e può venire decisa dopo qualche mese dall'acquisto del computer, quando e se vi renderete conto di averne davvero bisogno.

Un secondo disco da 5" 1/4 può essere necessario se vi trovate ad usare spesso programmi creati prima dell'avvento dell'UniDisk, programmi cioè che non permettono l'uso dei dischi da 800 Kbyte. Inoltre, per lo stesso motivo citato sopra, se avete la frequente necessità di effettuare copie dei dischi da 5" 1/4, un secondo drive è d'obbligo.

Per quanto riguarda il disco rigido, per quelli che hanno necessità di mantenere in memoria grandi quantità di dati, mi sento obbligato a fare una considerazione per aiutare il mio lettore ad usare al meglio il suo computer, anche se qualcuno potrebbe digrignare i denti.

Apple IIgs è una grande Macchina, con prospettive di crescita spettacolari dinanzi a sé: in quanto tale ha fame di memoria, come abbiamo già detto, e di periferiche al più alto livello di sofisticazione possibile.

Se decidete per l'acquisto di un disco rigido di grande capacità, il ProFile da 10 megabyte non è la scelta migliore. Se la decisione spettasse a me, io sceglierei di acquistare una interfaccia SCSI (Small Computer Systems Interface), la stessa che è incorporata nel Macintosh Plus, e un disco rigido da 20 (come quello Apple) o meglio ancora 30 o 40 Megabyte con uscita SCSI.

Un disco del genere costa un po' di più, ma ha capacità doppia o tripla del ProFile, può essere collegato anche da un Macintosh e, anzi, le due macchine possono usarlo contemporaneamente, e la stessa interfaccia SCSI che avete scelto vi sarà utile in futuro per collegamenti con altri computer.

6.7 Come funziona un disk drive

I più smaliziati dei miei lettori avranno scorso la prima parte di questo capitolo un po' annoiati, trovando tutte le notizie piuttosto risapute. Ecco qualcosa per i loro denti e per tutti i curiosi: una spiegazione dei principi sui quali si basano i dischi.

La prima cosa da fare per usare un disco è **formattarlo,** un brutto neologismo informatico che discende dall'inglese "to format", dare un formato.

Quando un disco viene formattato vengono create le tracce ed i settori: come abbiamo accennato poco sopra, le tracce sono anelli concentrici disposti sul disco, ed i settori sono segmenti di queste tracce. Su un disco 5" 1/4 ci sono trentacinque tracce da sedici settori da 256 byte ciascuno; su un disco 3" 1/2 ci sono 160 tracce con un numero variabile di settori (un numero maggiore sulle più ampie tracce esterne, un numero minore sulle tracce interne) da 512 byte ciascuno. I settori da 512 byte vengono normalmente chiamati **blocchi.**

I settori non sono contigui sulle tracce: tra l'uno e l'altro esistono degli spazi vuoti, chiamati **gap,** il cui scopo è duplice. In primo luogo la loro esistenza permette al programma che sta leggendo un disco di avere del tempo tra la lettura di un settore e la lettura del successivo, tempo dovuto all'attesa che la rotazione del disco lasci scorrere sotto la testina il gap e porti il settore successivo.

In secondo luogo, i gap servono da cuscinetto. I dischi son letti e scritti, e nulla vieta di reincidere un settore per aggiornare i contenuti: ma la velocità di rotazione, essendo meccanica e non elettronica, non è sempre esattamente la stessa. Se il drive rallenta un poco mentre la testina sta scrivendo un settore, il settore finisce per trovarsi scritto un po' prima di dove si trovava precedentemente; se il drive accelera, il settore risulterà spostato in avanti. Grazie ai gap, queste sbavature coprono soltanto un pezzetto del gap stesso: se il gap non esistesse si finirebbe per distruggere l'ultima parte del settore precedente o la prima parte del settore antecedente, con effetti disastrosi.

Un disco che conserverà dati o programmi deve anche venire **inizializzato.**

Lo spazio disponibile su disco, la sua capacità di memorizzare informazioni, non può essere destinata interamente alle necessità dell'utente. Uno spazio del disco deve registrare informazioni sullo stato del disco stesso; così come su un libro un indice composto di pagine viene usato per aiutare il lettore a trovare le informazioni che sta cercando, così su un disco un indice composto di settori riservati viene conservato ed aggiornato per tenere traccia delle informazioni che sono registrate.

Il formato in cui le informazioni sono scritte è differente tra i diversi sistemi operativi: per questo motivo un disco creato da DOS 3.3 non può venire usato da ProDOS nè da UCSD Pascal.

La fase di formattazione è identica per tutti i sistemi operativi, ed è differente da drive a drive (perché cambia il numero di tracce e di settori). La fase di inizializzazione è differente per ogni sistema operativo, ma identica per tutti i tipi di drive che un sistema operativo può usare.

Dal momento che ProDOS considera tutti i dischi composti da blocchi da 512 byte, d'ora in poi ci scorderemo dell'esistenza dei settori (i settori dei dischi 5" 1/4 vengono accoppiati, a due a due, per formare dei blocchi logici).
I blocchi vengono arbitrariamente numerati zero, uno, due... e così via sino ad una valore massimo che dipende dal tipo di disco.

Device	**Ultimo blocco (dec)**	**(hex)**
Disco 5" 1/4 da 140 Kbyte	279	0117
Disco 3" 1/2 da 800 Kbyte	1599	063F
Profile 5 MegaByte	5119	13FF
Profile 10 Megabyte	10239	27FF

Supponiamo che voi vogliate salvare una lettera che avete scritto ad un parente su un disco. Per voi la lettera è fatta da simboli alfabetici che compongono il testo; per il computer è fatta di numeri (ad ogni lettera corrisponde uno ed un solo numero, secondo un codice chiamato ASCII) che compongono il **file.**
Un programma è un file. Un testo scritto in italiano è un file. Anche l'immagine di un disegno è un file.
Se un file è composto di 1500 byte (per esempio se la lettera che avete scritto al vostro parente è composta da 1500 tra lettere dell'alfabeto, spazi, numeri e simboli) saranno necessari per memorizzarla 2 blocchi da 512 byte per intero. Un terzo blocco verrà occupato solo in parte, solo per 476 byte, ma sul disco quello spazio risulterà occupato per intero, perché il blocco è l'unità minima indivisibile di memoria che si può assegnare.
Da qualche parte bisognerà scrivere quali blocchi sono usati e quali sono occupati. Alcuni blocchi di ogni disco debbono allora conservare una **mappa a bit** del disco, dove per ogni blocco è indicato se è in uso per qualche file o se è libero e può essere assegnato ad un file che viene creato.

C'è un altro problema. Bisogna scrivere da qualche parte esattamente quali blocchi compongono il file, ed in che ordine vanno presi.

A queto scopo viene allocato un altro blocco, chiamato **blocco indice** in ProDOS e **Track Sector List** in DOS 3.3.
In Pascal si sceglie un approccio leggermente differente: un file deve essere composto da blocchi con numeri successivi (per esempio la vostra lettera potrebbe stare nei blocchi dal 18 al 20) ì dunque è sufficiente memorizzare il numero del primo blocco e la lunghezza del file in blocchi.
A questo punto, è necessario un indice generale di tutti i file sul disco, nel quale appare il nome del file e il numero del blocco indice oltre ad altre informazioni utili (per esempio la data del giorno in cui il file è stato creato). Un indice siffatto si chiama **directory.**

Dato che tutto quello che il drive fa è leggere o scrivere un blocco, tutte queste creazioni logiche chiamate file, directory, indice debbono essere curate a un programma speciale che gestisca i drive. Questo programma è il sistema operativo, l'argomento del prossimo capitolo. Ma prima di arrivarci...

6.8 Il lancio del sistema operativo

Problema, tanto semplice quanto sconvolgente: per usare il disco, il computer s rivolge al sistema operativo del disco (DOS). Ma il DOS si trova sul disco; così, come viene caricato in memoria dapprima? Sembra una riedizione tecnologica dell'uovo e la gallina: senza DOS non si usa il disco, senza usare il disco non si carica il DOS.

Ovviamente, la soluzione del problema sta nel nostro jolly, l'onnipresente System Monitor (del quale scopriremo i segreti del capitolo 8).
Monitor sa che deve cedère il controllo dell'Apple IIgs ad un sistema operativo, e gli da una mano per salire a bordo.

Monitor sa che un sistema operativo va caricato sul disco, e sa riconoscere la presenza di un disk drive: un disco dipende da uno slot, come tutte le periferiche collegate all'Apple IIgs. Ogni slot ha riservata per se la zona di memoria dalla locazione \$FF/Cn00 alla \$FF/CnFF, dove n è il numero dello slot. Un disco 5" 1/4 dipende dallo slot 6, ed ha quindi a disposizione lo spazio di memoria ROM da \$FF/C600 a \$FF/C6FF; un disco 3" 1/2 di-

pende dallo slot 5, ed ha a disposizione lo spazio di ROM da $FF/C500 a $FF/C5FF.

Monitor, quando Apple IIgs viene acceso o quando viene dato il comando di far partire il disco e caricare il sistema operativo, cede il controllo al programmino che si trova in quella zona di memoria, e che è a tutti gli effetti una sua parte.

Cosa accade a questo punto varia da sistema operativo a sistema operativo, e in qualche misura da drive a drive. Vediamo in caso di DOS 3.3 che viene fatto partire da un Drive 5.25 in Slot 6, dal disco in un loop, chiamando parti del Boot zero come sub-

Nel nostro caso, sappiamo che nello spazio $C600 va a risedere un piccolo programmino in grado di:
* Accendere il motore del drive.
* Spostare la testina sulla traccia zero (in un modo piuttosto brutale: impartisce l'ordine di ritirarla di *ottanta* tracce, mentre ne esistono solo trentacinque. Il risultato è quel tipico rumore di cliccheti-clak).
* Leggere il settore zero in memoria, nella locazione $0800.
* Eseguire un balzo incondizionato al programma testé caricato.

Chiameremo questo programma BOOT ZERO. E un po' l'equivalente del primo stadio di un razzo lanciato nello spazio: si limita a compiere la prima parte del lavoro – il decollo – poi si stacca e cade. Prima di terminare il suo lavoro ha caricato dal disco in memoria il primo pezzetto di DOS: il settore zero di traccia zero, contenete il programma che chiameremo BOOT UNO.

Boot uno è il secondo stadio: il suo lavoro consiste nel leggere dal disco in un loop, chiamando parti del Boot zero come suroutine, i settori dall'uno al nove della traccia zero. Finito il suo lavoro, anch'esso esegue un salto incondizionato all'inizio del programma caricato, il BOOT DUE.

Boot due è composto di due parti: un breve programma principale e la serie di subroutine integrate chiamate RWTS. È questa signora la vera protagonista assouta del nostro film. Il prossimo stadio consiste nel richiamare in memoria tutto il corpus del DOS, che risiede sul disco nel rimanente di traccia zero, nella traccia uno ed in parte della traccia due. Questo implica che dovremo spostare la testina all'esterno – ricordate, i settori di una traccia sono segmenti dello stesso cerchio, le tracce sono cerchi concentrici – e ciò complica notevolmente le operazioni necessarie. Per spostare la testina esattamente, i tempi vanno calibrati al milionesimo di secondo. RWTS può farcela, e lo fa sotto il con-

trollo di Boot Due; il DOS viene caricato ed il controllo gli viene passato.

Nel caso ve lo stiate chiedendo, il termine "Boot" deriva dall'inglese "bootstrapping", che indica il concetto di "allacciarsi gli stivali sollevando entrambi i piedi all'altezza delle mani", il che è tanto impossibile quanto l'idea di usare il DOS per caricare il DOS.

7

IL SISTEMA OPERATIVO

7.1 I sistemi operativi della famiglia Apple II

Prima di parlare del sistema operativo creato specificatamente per sfruttare al massimo le potenzialità dell'Apple IIgs, il ProDOS 16, vale la pena di parlare brevemente dei sistemi operativi creati per i modelli precedenti della serie e di come si comportano su Phoenix.

Andiamo in ordine cronologico di apparizione, e citiamoli tutti almeno in breve.

Notate che non è mia intenzione spiegarvi come funzionino tutti questi sistemi operativi: richiederebbe troppo spazio. A ciascuno corrispondono alcuni manuali della Apple Computer che sono sin troppo esaurienti nello spiegarne le caratteristiche.

Lo scopo qui è spiegare come dovrete aspettarvi che questi sistemi operino su un Apple IIgs, un computer che non era ancora disponibile quando sono stati creati, e aggiungere per buona miura qualche trucco che può rendere più facile la vita con essi.

È ovvio che le piene potenzialità dell'Apple IIgs non vengono usate da nessuno di questi sistemi: Phoenix si trova ad operare emulando uno dei modelli precedenti, e tutte le sue capacità sono sottostimate.

I sistemi operativi sono divisi in quattro famiglie. Un membro di ciascuna famiglia può utilizzare i dischi creati da un altro

membro e non può usare i dischi di ogni altro sistema operativo. È sempre conveniente usare la versione più recente disponibile di un sistema operativo, che sarà in grado di sfruttare meglio delle versioni precedenti le capacità del computer. A maggior ragione questo vale con ProDOS: dovreste sempre avviare le applicazioni ProDOS con ProDOS 16 installato nel IIgs.

Delle famiglie di sistemi operativi fanno parte diverse **versioni**. Spesso mi sono sentito chiedere cosa significhino quei numeri e come vengano attribuiti, e dunque apro una parentesi a questo punto per spiegarlo...

Il numero di versione identifica una *release*, cioè lo stato di un qualunque programma distribuito, di solito a pagamento oppure gratuitamente, come nel caso dei sistemi operativi.

Quando un programma viene creato da un programmatore gli viene convenzionalmente dato il numero 1.0 (che si legge "uno punto zero").

Può darsi che dopo il momento in cui il programma viene diffuso venga scoperto un piccolo errore di programmazione, che lo fa comportare anormalmente in qualche circostanza. Per esempio, il primissimo ProDOS (il ProDOS 8 versione 1.0) si dimenticava di controllare se il disco inserito nel drive fosse stato o meno protetto contro la scrittura dall'utente, causando prestazioni blandamente irritanti.

In quel caso l'errore (il bug come lo chiamano i programmatori, o *baco* se ne preferite una approssimativa traduzione italiana) viene corretto, e la nuova versione viene diffusa come 1.0.1.

Ma ammettiamo che si renda necessario un cambiamento più drastico nel programma. Prendiamo per esempio il caso del Pascal 1.2, che non era in grado di usare i dischetti da tre pollici e mezzo che contengono 800 Kbyte. In quel caso si rende necessario riscrivere una parte del programma, e ancora una volta il numero di versione viene modificato.

Pascal 1.2 divenne Pascal 1.3, che può usare i Drive 3.5. In modo analogo da ProDOS 1.0.2 si è passati al ProDOS 1.1.

Può anche darsi che un programma venga totalmente riscritto, per migliorarne le prestazioni, per aggiungere nuove capacità, per adattarlo a un nuovo computer o a un nuovo sistema operativo. In questo caso si parla di "versione differente", ed il numero di versione cambia ancora. In questo modo abbiamo avuto il passaggio da Apple Writer II 1.0 ad Apple Writer II 2.0.

Le stesse regole si applicano per i numeri di versione dei sistemi operativi, che sono anch'essi dei programmi, molto complessi.

Esistono delle utilità (sarebbe meglio dire "dei programmi di sistema" ma mi adeguo al modo di dire) che permettono di operare indistintamente su più sistemi operativi.
Il programma chiamato "Utilities di sistema" e distribuito insieme agli Apple IIc e agli Apple IIe più recenti permette di trasferire dei file da un sistema operativo ad un altro, così come fa un ottimo prodotto chiamato "Universal File Converter" che generalmente reputo più sicuro – mentre il primo non è affatto a prova di bomba.
Più modestamente il copiatore chiamato "Copy II Plus" nelle versioni 6.X e 7.X permette di svolgere il trasferimento di file più comune, e cioè da DOS a ProDOS e viceversa. Mi permetto di consigliarvelo, in quanto è ben fatto, semplice da usare e dispone di numerose altre opzioni spesso utili.

Le quattro famiglie dei sistemi operativi della famiglia II sono:

* DOS 3.0, DOS 3.1, DOS 3.2, DOS 3.2.1
* DOS 3.3 (e le sue variazioni non ufficiali)
* Pascal 1.X (con X un qualche numero)
* ProDOS (distinto in due rami, ProDOS 8 e ProDOS 16)

Sugli Apple II+, IIe e sul nostro IIgs è anche possibile usare un sistema operativo, il **CP/M**, piuttosto diffuso. CP/M richiede che in uno slot del sistema venga inserita una scheda contenente un microprocessore, lo Z80, che si sostituisce al microprocessore dell'Apple II quando viene usato il CP/M stesso.
Dato che esistono numerose versioni della scheda e del sistema operativo, e dato che non è disponibile su ogni Apple II ma solo su quelli opportunamente modificati, non ne parlerò p'ù.

7.1.1 Il DOS 3.0, 3.1, 3.2, 3.2.1

C'era una volta, molto molto tempo fa, la Apple Computer, e c'era una volta l'Apple II. Era un grande computer, con 4 kilobyte di RAM – ma poteva essere espanso sino a 16!
I programmi venivano salvati sulle cassette, e tutti erano molto infelici, perché le cassette sono molto lente.
Un giorno un prode eroe (tale Steve Wozniak, come avrete già supposto), entrò nel suo laboratorio, e lavorò, e lavorò, e lavorò...

Il 29 giugno 1978 Apple presentò il disk drive per Apple II, il DISK II, ed il sistema operativo che lo accompagnava, chiamato

DOS 3.0 (se vi interessa, DOS 1 e DOS 2 furono utilizzati solo internamente alla Apple Computer, e non sono ricordati dalle cronache dell'epoca). Ma la fretta aveva giocato un gran brutto scherzo ai Nostri: il sistema operativo era tutto un baco, pieno di errori, quasi inutilizzabile; fu così che per il 20 luglio Apple distribuì DOS 3.1, funzionante. Qualche mese dopo, il 16 febbraio 1979, fu la volta di DOS 3.2, che migliorava notevolmente alcuni aspetti del trattamento dei file, e il 31 luglio dello stesso anno giunse DOS 3.2.1 che fissava un paio di errori trascurabili della versione precedente.

Apple IIgs è in grado di utilizzare persino questi sistemi antidiluviani – ammettendo che voi riusciate a rintracciare qualche dischetto da 5 pollici ed un quarto organizzato in questo modo.
C'è comunque bisogno di un espediente. I minidischi di Apple II – come quelli del Commodore 64, per esempio – hanno 35 "tracce", ovvero sono divisi in 35 solchi circolari e concentrici; sotto questi primitivi sistemi operativi ciascuna traccia era ulteriormente suddivisa in 13 "settori". Un settore registra 256 byte di informazioni (il settore è l'unità minima di informazione che potete leggere o scrivere da un dischetto cinque pollici – ne abbiamo parlato nello scorso capitolo).
Tutti i sistemi operativi successivi hanno introdotto un miglioramento, usando sedici settori per traccia, permettendo un aumento di tre sedicesimi nel quantitativo del materiale che un disco può memorizzare, cioè circa il 18,75%.
Di conseguenza non è possibile utilizzare – lanciandolo – direttamente un disco in formato tredici settori. È invece necessario eseguire per primo un programma chiamato **BOOT13** che trovate sul **DOS 3.3 System Master Diskette**, un disco da cinque pollici che veniva venduto insieme ad ogni Disk II sino all'avvento di ProDOS, l'attuale sistema operativo. Potete acquistare una copia del Dos 3.3 System Master in un Apple Center o più semplicemente chiederne una copia ad un amico. Lanciate quel disco e quando siete di fronte al segnale di pronto del Basic, la parentesi quadra chiusa, comandate **BRUN BOOT13**.

Se utilizzate uno dei primi DOS il vostro computer verrà visto come un Apple II+ con 48 Kbyte di RAM e 12 Kbyte di ROM. Dovete tenere abbassato il tasto blocca maiuscole, perché quel computer usava esclusivamente le maiuscole. I vostri Disk 3.5 non possono essere utilizzati, e i dischi da cinque pollici non possono essere letti a meno che siano stati creati sotto quello stesso DOS.

7.1.2 Pascal 1.0

Per introdurre il linguaggio Pascal sul nostro amico fu necessario scrivere un nuovo sistema operativo (UCSD–Pascal), e i geniacci sotto il comando di Steve Wozniak ne approfittarono per introdurre una modifica nello hardware del disco.

Con Pascal, si migliorò lo hardware dell'interfaccia disco permettendo l'introduzione di 16 settori per traccia, portando così la capacità del disco da circa 115 kilobyte per facciata a circa 140 – a quanto mi risulta, l'unico punto a favore di Pascal in tutta la sua storia.

Disgraziatamente, UCSD Pascal 1.0 venne creato con tecniche a dir poco approssimative: infatti quel sistema operativo fa dei riferimenti assoluti alle locazioni interne del computer. Per dirla in termini semplici, si aspetta di essere usato su un Apple II+, e non può funzionare su nessuno dei modelli più recenti.

Se avete assoluta necessità di usare un programma sviluppato sotto Pascal 1.0 procuratevi una versione più recente del sistema (la più recente in assoluto mentre scrivo è la versione 1.3, ma non dubito che presto ne uscirà una nuova). Poi utilizzate il **Filer** per trasferire tutto quanto si trova sul disco Pascal 1.0 su un disco appena inizializzato, **con l'eccezione dei file chiamati SYSTEM.APPLE e SYSTEM.PASCAL e SYSTEM.MISCINFO**. Copiate invece questi tre file dalla vostra copia della versione più recente. Dovreste così ottenere un disco eseguibile da Apple IIgs.

7.1.3 DOS 3.3

Anche DOS 3.2 venne aggiornato per poter usufruire dei sedici settori per traccia, dando così origine al DOS 3.3: era il 25 agosto 1980.

Il regno di DOS 3.3 fu lungo e felice, più di tre anni, sinché non giunse nelle nostre contrade ProDOS, l'orgoglioso.

Potete usare perfettamente DOS 3.3 sul vostro Apple IIgs. Quando usate DOS 3.3 da Basic dovete avere l'accortezza di impartire tutti gli ordini usando esclusivamente le lettere maiuscole.

Con DOS 3.3 installato ed in controllo della macchina, non potete utilizzare i disk 3.5, ma solo i disk 5 pollici. Il sistema riconosce solo i primi 64 Kbyte di Ram e 16 Kbyte di ROM.

Una ditta privata, la MicroSpark, ha creato una versione adattata di DOS 3.3 in grado di utilizzare anche i Disk 3.5, battezzandola UniDOS – dal nome degli UniDisk. Tuttavia non si tratta di un programma ben curato, e la maggior parte dei programmi basati su DOS non funzioneranno se passati all'UniDOS; quasi esclusivamente i programmi Basic possono beneficiare della modifica. Decisamente meglio funziona la seconda versione di UniDOS, battezzata UniDOS Plus.

7.1.4 Pascal 1.1

Pascal 1.1 è la più vecchia versione del sistema operativo UCSD–Pascal in grado di funzionare sull'Apple IIgs.

Pascal 1.1, come DOS 3.3, crederà che il vostro sia un Apple II+, quindi per utilizzarlo sarete tenuti ad usare unicamente le lettere maiuscole ed a dimenticare i Disk 3.5.

Nell'Editore di programmi potete usare le lettere minuscole. Verranno visualizzate sullo schermo come maiuscole *inverse*, cioè nero su bianco anziché bianco su nero, ma verranno eventualmente stampate correttamente come lettere minuscole da una stampante se richiederete la stampa del testo.

Il linguaggio Fortran venduto da Apple si basa su Pascal 1.1, e pertanto il suo uso è difficoltoso su un Apple IIgs. Purtroppo, il disco è protetto contro la copia, ragion per cui è impossibile copiarne le parti fondamentali su un disco che lavori con una versione più recente di Pascal. Credo e spero che mamma Apple avrà il buon gusto di adattare Fortran al più presto.

Notate che in tutte le versioni del sistema operativo Pascal viene attivato il display ad ottanta colonne al lancio, e non è possibile evitarlo (le istruzioni del Pannello di Controllo vengono trascurate). Sull'Apple IIc esiste un pulsante che serve a selezionare 40 oppure 80 colonne, ed i Pascal più recenti (dalla versione 1.2 in poi) leggono lo stato del pulsante prima di decidere se debbono davvero attivare il modo ottanta colonne.

L'incapacità del Pascal a riconoscere lo stato del Pannello di Controllo come sinonimo di quel pulsante è uno dei motivi che mi fanno pensare al rilascio di una versione ulteriore come molto probabile.

7.1.5 Prodos 8 Versioni 1.0 ed 1.1

Era l'ottobre del 1983 quando DOS 3.3 venne sostituito da ProDOS, ma ancora una volta la fretta aveva giocato un brutto scherzo a Mamma Apple: così si ebbe ProDOS 1.0.1 il 1 gennaio 1984, e ProDOS 1.0.2 il 15 febbraio 1984. Dopo pochi mesi, ecco ProDOS 1.1.1.
Per quel che ci riguarda, tutte queste versioni si comportano identicamente su un Apple IIgs.

Spendiamo ora qualche parola per elencare le differenze macroscopiche tra il vecchio DOS 3.3 ed il ProDOS.

DOS può usare solo il DISK II, ProDOS 8 può usare anche i Disk 3.5 ed il PROFILE, il disco rigido della Apple (può usare dischi sino a 32 megabyte di capacità).
ProDOS può avere nella directory principale 52 file, DOS 3.3 può averne 105 e i precedenti 84... però ProDOS ammette le subdirectory, sottoindici che possono custodire qualsiasi numero di file, in cambio dell'occupazione di ulteriore spazio sul disco.
DOS ammette titoli di file sino a 30 caratteri di qualunque tipo (lettere, numeri, spazi, simboli, caratteri di controllo), ProDOS accetta un massimo di 15 caratteri, e solo lettere, numeri e punti. Per esempio, "IO & TE= NOI 2" è un nome legale per DOS e illegale per ProDOS che userebbe IO.TE.NOI2.
DOS funziona su qualunque Apple II che abbia almeno 16 Kbyte di Ram, e sia con Integer Basic che con Applesoft Basic in ROM. ProDOS funziona solo con gli Apple II da 64 kilobyte (o più, ma non gestisce direttamente quelli aggiuntivi), e solo con Applesoft.

ProDOS 8 può venire eseguito su un Apple IIgs, e lo considererà come un Apple IIe. Sotto ProDOS veng no utilizzati 128 Kbyte di RAM, ed è possibile usare pienamente tutto il set di caratteri della tastiera, impartendo comandi indifferentemente con lettere maiuscole o minuscole.
I 128 Kbyte di RAM sono visti in due blocchi di 64K di cui il primo è sempre in uso mentre il secondo viene concesso alle applicazioni che lo richiedono (come Apple Works, Mouse Desk, Instant Pascal per citare i più diffusi) o, se non è richiesto da programmi simili, viene usato come disco Ram, chiamato /RAM, collegato da un punto di vista logico allo Slot 3, Drive 2. Questo disco Ram non va confuso con il disco /RAM5: il disco RAM selezionato da Pannello di Controllo, chiamato /RAM5.

ProDOS aggiunge data ed ora di creazione e modifica ai file: purtroppo l'orologio di Apple IIgs non rispetta le convenzioni fissate dalla Apple per l'uso con i primi ProDOS, e dunque non viene consultato.

7.1.6 Pascal 1.2

Con Pascal 1.2 è possibile usare sino a 128 Kbyte di RAM, ma non sono ancora disponibili i dischi 3.5.

L'uso delle lettere minuscole non comporta difficoltà: Pascal 1.2 è stato creato per l'uso sugli Apple IIe e IIc che possiedono, come il IIgs, una tastiera completa – a differenza dei modelli precedenti.

A differenza di ProDOS, Pascal non permette l'uso di 64 Kbyte come disco RAM: tutti i 128 Kbyte visibili sono occupati dai programmi. Questo comporta la perdita di una comodità come il disco RAM ma permette la stesura di programmi anche parecchio lunghi, più di quanto sia possibile fare con ProDOS in Basic (soli 38 Kbyte). In effetti, la capacità di Pascal di usare i 128 Kbyte degli Apple IIe e IIc come un unico blocco di memoria è unica – naturalmente su un Apple IIgs l'uso della memoria è molto meno problematico e tutti i nuovi linguaggi creati per essere usati su questa macchina possono usare tutta la memoria di cui hanno bisogno senza problemi.

Sempre a differenza di ProDOS, Pascal non è in grado di distinguere automaticamente che ci sono più di 64 Kbyte disponibili. Se laciando i dischi di Pascal 1.2 appare la scritta *Pascal System size is 64K* sarà necessario fare qualche manipolazione.

Il pascal 1.2 era fornito su quattro dischetti da cinque pollici, chiamati APPLE0: APPLE1: APPLE2: APPLE3: (il due punti finale indica che si tratta di dischi creati con Pascal, così come i dischi ProDOS hanno il nome preceduto dalla sbarra, come in /APPLEWORKS. I dischi DOS 3.3 non hanno un nome ma solo un numero, cosiddetto numero di volume).

Sul disco APPLE3: si trovano due file, chiamati 128K.APPLE e 128K.PASCAL che devono essere copiati sul disco di lancio, che sarà APPLE0: se possedete un solo disco da cinque pollici, e sarà APPLE1: se ne avete almeno due. Usando il Filer di Pascal copiate il file APPLE3:128.APPLE sul disco di lancio con il nome SYSTEM.APPLE (Filer chiederà se volete distruggere il file che si trova su quel disco con lo stesso nome, rispondete di sì). Copiate quindi il file 128K.PASCAL dandogli il nome di SYSTEM.PASCAL.

7.1.7 Pascal 1.3

Pascal 1.3 non presenta molte differenze dalla versione precedente del sistema Pascal, e la maggior parte delle osservazioni fatte per quello valgono anche per questo.

Pascal 1.3 permette l'uso degli Apple Drive 3.5: anche per i dischi da tre pollici vale la considerazione che il formato è differente da quello ProDOS, e dunque non è possibile far leggere i dischi ProDOS a Pascal e viceversa.

Sotto Pascal 1.3 è possibile utilizzare il disco RAM creato da Pannello di Controllo. Se avete lanciato il Pascal come prima cosa all'accensione dell'Apple IIgs il disco sarà già disponibile. In caso contrario, il disco RAM viene visto come un disco ProDOS e non può essere usato con Pascal.

È possibile usare comunque il disco RAM, ma ogni dato che vi sia stato copiato sotto ProDOS andrà perso! Se non c'è nulla di importante sul disco RAM entrate nel Filer di Pascal, togliete tutti i dischi non Pascal dai vostri drive e chiedete di leggere i nomi dei Volumi in linea.

Noterete che viene elencato un disco come <no dir>, cioè senza directory Pascal: è il disco RAM. Osservate attentamente il numero di volume che viene presentato (probabilmente sarà #5 oppure #12).

A questo punto battete Z, il comando di Filer per azzerare un disco. Rispondete alla domanda battendo il simbolo diesis o cancelletto che dir si voglia e il numero che avete trovato. Confermate la scelta e scegliete un nome per il disco (RAM: andrà benissimo). Avete a vostra disposizione il disco RAM.

7.2 Il ProDOS

Siamo arrivati a parlare dei sistemi operativi contemporanei dell'Apple IIgs, che riescono dunque a riconoscerlo e ad utilizzarne tutte le potenzialità.

ProDOS è l'attuale sistema operativo ufficiale degli Apple II. È un sistema completo, ben fatto, potente e veloce, anche se alcuni dei programmi al corredo non sono altrettanto degni (è il caso del Basic.System, il programma usato per programmare in Basic Applesoft).

Data la sua importanza, gli dedico la parte restante del capitolo, dove ne spiegherò il funzionamento anche nei dettagli interni, ad uso e consumo degli inguaribili curiosi.

Attualmente ProDOS esiste in due versioni differenti: il ProDOS 8 ed il ProDOS 16. Mentre scrivo le ultime versioni disponibili hanno rispettivamente i numeri di sviluppo 1.3 e 1.1.
La differenza tra i due sta nel fatto che il ProDOS 8 può essere eseguito sugli Apple II dotati di microprocessore 65C02, mentre il ProDOS 16 è riservato all'Apple IIgs con il suo 65C816: i numeri 8 e 16 non sono numeri di versione, ma si riferiscono al fatto che il primo processore è un "otto bit" ed il secondo è un "sedici bit".
Poiché vengono prodotti sia degli Apple II con processore a 8 bit (gli Apple IIe ed Apple IIc) sia con 16 bit (lo Apple IIgs), entrambi i sistemi operativi vengono aggiornati quando ve ne sia necessità. Il ProDOS 16 comprende in sé il ProDOS 8, e gli delega i compiti più semplici così come la gestione dei programmi creati per funzionare anche sugli Apple IIe e IIc che lo Apple IIgs esegue in emulazione.

Ecco alcuni termini che potreste trovare nel seguito del capitolo e nella letteratura tecnica concernente gli Apple II...

Un **programma di sistema** è un programma richiede direttamente a ProDOS l'uso della memoria e dei dischi, attraverso un meccanismo di comunicazione che spiegherò tra poco. I programmi di sistema sono scritti in Assembler, il linguaggio nativo dell'Apple, da programmatori molto in gamba.
I programmi di sistema vengono riconosciuti dal loro **tipo**. Quando ne guardate i simboli in Mouse Desk, sono raffigurati da una mano che scrive, quando chiedete un CAT da Basic Applesoft sono identificati dalla scritta SYS oppure S16.
I programmi di sistema di ProDOS 8 hanno di solito il nome che termina per .SYSTEM, e quelli di ProDOS 16 hanno il nome che finisce con .SYS16: questo significa che se ProDOS 16 si vede arrivare la richiesta di eseguire un programma che si chiama, per esempio, APLWORKS.SYSTEM non lo fa direttamente ma passa la richiesta al ProDOS 8.

Non tutti i programmi sono programmi di sistema. Alcuni, come i programmi Basic che voi potete scrivere, non lo sono.
In quel caso il vostro programma viene eseguito dall'Apple II, e le sue richieste vengono passate a ProDOS, attraverso la mediazione di un programma di sistema. Il programma di sistema

che usate insieme ai vostri programmi Basic si chiama **Basic.System.**

Potete pensare ad un disco, qualunque sia il suo formato fisico, come ad un **volume** che contiene svariati documenti. Sia il volume che i file hanno dei nomi, compisti di lettere numeri e punti; il nome del volume viene tradizionalmente fatto precedere dal simbolo di sbarra per distinguerlo: ad esempio il volume /IL.MIO.DISCO può contenere il programma BASIC.SYSTEM, il programma DESKTOP.SYS16, il documento LA.LETTERA.N2. Tutti questi prendono il nome collettivo di **file**.

Esiste un tipo speciale di file che prende il nome di **subdirectory** o più brevemente di **directory** (qualcuno lo chiama "direttorio" italianizzando il nome, ma al sottoscritto quel termine fa proprio un po' schifo).
La subdirectory non è un documento contenuto nel volume, ma piuttosto una cartelletta che contiene a sua volta più documenti, raggruppandoli per mantenere ordine.
Una directory può, naturalmente, contenere a sua volta una subdirectory. Non tutti i sistemi operativi permettono l'uso di subdirectory dentro i volumi: dei sistemi operativi disponibili sugli Apple II, solo ProDOS lo consente.

Il sistema operativo è anch'esso un programma: il file del sistema operativo si chiama PRODOS, e potete trovarlo in tutti i dischi che, lanciati, eseguono un programma. Dove c'è ProDOS (ProDOS 8 versione 1.2 oppure ProDOS 16) c'è anche una subdirectory che si chiama SYSTEM, che contiene alcuni file usati dal ProDOS. Ci arriviamo nel prossimo paragrafo...

7.3 Cosa succede quando lanciamo un disco

Quando lanciate un disco con ProDOS, la ROM di Apple II carica il blocco zero in memoria (se state usando un Disk II il blocco viene caricato in due metà).
Il blocco zero (il **loader**) cerca nella directory principale del volume (non nelle subrdirectory) un file chiamato PRODOS: se non lo trova blocca le operazioni visualizzando il messaggio ***UNABLE TO LOAD PRODOS*** (incapace di caricare

ProDOS). In questo modo, i dischi dati che non hanno bisogno di contenere una copia del sistema operativo e possono così guadagnare spazio, segnalano la loro incapacità di governare la macchina.

Se c'è, il file PRODOS viene caricato in memoria all'indirizzo $2000. La sua prima parte, che prende il nome di **relocator** viene allora eseguita, e si mette come un pazzo a fare ogni tipo di lavori sporchi. Controlla che voi abbiate almeno 64K di RAM, stampa il copyright di Apple Computers e il suo numero di serie sullo schermo, controlla se c'è una scheda orologio, fa una serie di test per scoprire quale modello di Apple II sta eseguendo ProDOS, determina esattamente quanta memoria c'è installata, convalida l'esistenza del disco /RAM.

A questo punto, scatta una seconda fase. Il corpo vero e proprio del ProDOS non è stato ancora caricato in memoria: si trova infatti in una subdirectory che si chiama System (la stessa cosa vale esattamente identica su un Apple Macintosh, anche se il sistema operativo è ovviamente diverso).

Se il relocator ha scoperto di trovarsi su un Apple IIe oppure IIc carica in memoria il file P8, che contiene il ProDOS 8. Se invece tutto il processo ha avuto luogo su un Apple IIgs viene caricato P16, il ProDOS 16, e solo in secondo luogo il ProDOS 8.

Il nome di relocator deriva dal fatto che il ProDOS non viene caricato in memoria nel punto esatto in cui risiederà durante l'uso della macchina – e questo perché non sappiamo esattamente **quanta** memoria la macchina abbia al momento del caricamento. Il ProDOS viene dunque spostato nella memoria (e in gergo tecnico informatico si dice che viene **rilocato**) sino alla sua sede definitiva.

A questo punto, ProDOS ha preso possesso di Apple II.

ProDOS 8 passa a cercare sul disco che lo ha lanciato un file il cui nome finisca per ".SYSTEM" e che sia di tipo SYS, mentre se il controllo delle operazioni spetta al ProDOS 16 viene cercato un file il cui nome finisca per ".SYS16". Lo trova, lo carica, lo esegue: il suo compito per ora è terminato.

A questo punto, qualunque programma può essere eseguito.

7.4 Dentro ProDOS

Come DOS 3.3, anche ProDOS è uno e trino. È composto infatti da MLI **Machine Language Interface**, un sistema flessibile e completo per impartire richieste al sistema (come "leggi un file", "dimmi che ore sono" oppure "dimmi come si chiama il disco che sto usando"). Il **Kernel** esegue i comandi impartiti alla MLI: poiché i comandi sono identici per mezzi diversi (cioè MLI accetta lo stesso comando per aprire un file indifferentemente sul disco RAM, sul disco 5", sul minidisco 3" e sullo hard disk), Kernel deve identificare il tipo di mezzo in uso e compiere l'operazione sfruttando l'opportuno **device driver**.

I device driver sono routine fornite a ProDOS che gestiscono uno specifico disco o scheda: programmi in linguaggio macchina che debbono accettare certe convenzioni fissate da ProDOS e che debbono fornire certi risultati.

Per esempio, si richiede che il driver di un disco sappia fare quattro operazioni: formattare il disco, scrivere un blocco, leggere un blocco, dire se l'unità è pronta a fare un lavoro.

ProDOS contiene in se il device driver per il buon, vecchio Disk II da 140 K e per il disco RAM: gli altri driver stanno in ROM (Unidisk e Profile compresi) o vanno installati dentro ProDOS secondo certi metodi stabiliti.

I device driver per le stampanti e per l'uso della rete di comunicazione AppleTalk non hanno bisogno di trovarsi dentro ProDOS, e vengono custoditi in una subdirectory della directory System, che ha il nome di Drivers, e vengono caricati in memoria per essere usati quando ce n'è bisogno.

7.5 Uso di ProDOS

ProDOS è costruito in modo da poter eseguire per conto dei vostri programmi tutti i compiti che richiedono un uso del disco o l'assegnazione di memoria. I programmi di sistema richiedono delle prestazioni a ProDOS tramite le **chiamate di sistema**; da un punto di vista logico, infatti, si tratta di chiamare il sistema operativo notificandogli la richiesta secondo un protocollo (cioè ri-

spettando certe convenzioni che la Apple ha pubblicato nel ProDOS Technical Manual).

Ad esempio, vediamo cosa succede dentro la macchina quando voi decidete di usare Mouse Desk per scoprire cosa contenga un disco.

Inserite il disco Mouse Desk in un drive ed accendete la macchina. Questo comporta la sequenza di operazioni descritte nel paragrafo 7.3, che termina quando ProDOS è installato in memoria e Mouse Desk è funzionante.

A questo punto voi togliete il disco Mouse Desk dal suo drive 3.5: cosa accade?

Mouse Desk richiede, ogni pochi secondi, a ProDOS di segnalargli se qualche disco è stato inserito o tolto dai drive (è la chiamata *status*). ProDOS esegue il compito con una chiamata ai device drivers dei dischi, riuniti e gestiti sotto il nome di SmartPort, chiedendo se sono pronti ad eseguire un lavoro: se SmartPort, dopo aver interrogato lo hardware del disco secondo modi che variano da disco a disco, segnala che il disco non è disponibile mentre lo era alla richiesta precedente, Mouse Desk può dedurne che il disco è stato tolto. Di conseguenza, fa sparire l'icona del disco dalla scrivania.

Voi posate il disco Mouse Desk e prendete il disco sconosciuto, poi lo infilate nel drive che avete liberato. In quel breve periodo, Mouse Desk ha già chiesto a ProDOS almeno centomila volte se c'è qualche novità!

Mouse Desk, grazie al principio che abbiamo spiegato poco fa, viene informato che un disco è stato messo nel drive.

A questo punto Mouse Desk fa una chiamata a ProDOS che va sotto il nome di ONLINE, alla quale ProDOS deve rispondere Si o No, a seconda che sia riuscito o meno ad indentificare il disco. Se la risposta è Si, ProDOS è anche tenuto a comunicare il nome del disco a Mouse Desk, scrivendone il nome in un punto della memoria che Mouse Desk gli ha segnalato (un **buffer**).

La Machine Language Interface di ProDOS riceve la richiesta, la esamina, la convalida. Passa la richiesta al Kernel. Il Kernel ordina al Device Driver di quel disco di leggere il blocco due del disco: su quel blocco si trova infatti il nome del disco.

Il Device Driver esegue la richiesta. Il Kernel esamina i dati letti e confronta certi punti chiave con i valori che dovrebbero avere se il disco fosse un disco creato da ProDOS.

Supponiamo che non lo sia. Il disco sconosciuto è un disco Pa-

scal. In questo caso, Kernel si prepara a rispondere che non si può usare quel disco (risponde di No alla chiamata di Mouse Desk). Lo fa, (questo vuole dire qualcosa per voi solo se conoscete il linguaggio Assembler), ponendo ad 1 il flag di carry.

Il Kernel, il cuore di ProDOS, termina il suo compito. La chiamata è stata esaudita, ed il controllo degli affari torna a Mouse Desk. Mouse Desk si accorge che il disco è anomalo, visto che ProDOS gli ha detto di no. Può allora far apparire la finestra con la scritta "Attenzione! Disco non ProDOS!".

Supponiamo ora che il disco fosse davvero un disco ProDOS, e per l'esattezza un disco che contiene una copia di Apple Works.

Il Kernel effettua i suoi controlli di valore, e si rende conto che il disco è ProDOS. In tutti i dischi ProDOS il nome del volume è registrato in un certo punto del blocco due: Kernel può copiare quei valori (le lettere che compongono il nome APPLEWORKS) nel punto della memoria che Mouse Desk gli ha comunicato facendo la chiamata. Poi si prepara a rispondere di Si e torna il controllo a Mouse Desk, il quale fa apparire sullo schermo l'icona del disco e ci scrive sotto il nome che ProDOS gli ha passato, il nome del vostro disco, "Appleworks".

Per i programmatori che conoscono il linguaggio macchina aggiungo che le chiamate a ProDOS funzionano così.

La chiamata ha la forma di un Jump to Sub Routine all'indirizzo assoluto $00/BF00, una locazione di memoria chiamata MLI ENTRY. Di seguito al JSR $BF00 si trova un byte con un codice corrispondente all'operazione richiesta, e poi una word con l'indirizzo alla zona di memoria che contiene i parametri (un puntatore allo spazio in cui copiare il nome del volume in linea, per rifarci all'ultimo esempio).

Eccovi, per curiosità, una tabella con i codici delle chiamate possibili sotto ProDOS 1.2

Codice	Nome	Scopo
$40	AllocInterrupt	Segnala la presenza di un gestore di interrupt
$41	DeallocInterrupt	Rimuovi lo *interrupt handler*
$65	Quit	Chiama il Filer o Mouse Desk
$80	ReadBlock	Leggi un blocco da disco
$81	WriteBlock	Scrivi un blocco su disco
$82	GetTime	Consulta l'orologio per avere data ed ora

(Segue)

(Seguito)

$C0	Create	Crea un file nella directory attuale
$C1	Destroy	Cancella un file
$C2	Rename	Cambia nome a un file
$C3	SetFileInfo	Blocca, sblocca, aggiorna dati su un file
$C4	GetFileInfo	Ottiene i dati vitali di un file (lunghezza etc)
$C5	OnLine	Ottiene i nomi dei dischi nei drive
$C6	SetPrefix	Specifica una diversa directory da usare
$C7	GetPrefix	Ottiene il nome della attuale directory in uso
$C8	Open	Prepara un file a lettura o scrittura
$C9	NewLine	Seleziona o deseleziona il carattere di a capo
$CA	Read	Leggi dati da un file
$CB	Write	Scrivi su un file
$CC	Close	Chiudi il file
$CD	Flush	Forza l'aggiornamento di un file
$CE	SetMark	Specifica un punto nel file dove operare
$CF	GetMark	Ottiene la lunghezza del file
$DO	SetEOF	Specifica la lunghezza del file
$D1	GetEOF	Ottiene la lunghezza di un file
$D2	SetBuf	Sposta il buffer per un file aperto
$D3	GetBuf	Riconosce la zona di memoria usata da un file
	ClearBackupBit	Segnala che un file è stato ricopiato
	GetBootVol	Specifica il nome del disco di lancio del sistema
	GetVersion	Restituisce il suo numero di sviluppo (1.2 o più)
	GetDeviceNum	Comunica slot e drive di una periferica

7.6 SmartPort

Con il secondo modello di Apple IIc (vedi appendice A), Apple ha introdotto un'interessantissima miglioria alla gestione dei dischi. La miglioria, il protocollo intelligente chiamato SmartPort, è stato mantenuto e migliorato su Apple IIgs.

I dischi si dividono in due grandi gruppi: quelli che permettono la lettura dei dati per blocchi di grandezza data e quelli che la

consentono byte per byte. Tutti i drive Apple che abbiamo visto in questo capitolo sono del primo tipo, ma nulla vieta che in futuro ne vengano introdotti del secondo tipo. Altre distinzioni sono quelle tra dischi intelligenti (cioè che contengono un microprocessore dedicato e lo usano per ottimizzare lo scambio di informazioni col computer) e non. Tra i dischi Apple, l'unico intelligente è UniDisk, che contiene un 65C02.

Altra distinzione: tra i dischi che richiedono l'attenzione del processore del computer per leggere e scrivere (tutti i dischi che abbiamo visto agiscono in questo modo) e dischi che possono accedere direttamente alla memoria del computer senza intervento del 65816, chiamati anche dischi DMA (**direct memory access**). Di questi ultimi ci occuperemo ancora nel capitolo 10.

Grazie all'introduzione di SmartPort, oggi è possibile per un costruttore di hardware, sia esso la Apple Computer o un produttore indipendente, sviluppare un tipo qualunque di disk drive e collegarlo ad Apple IIgs sotto ProDOS senza preoccuparsi di problemi software.

SmartPort (il cui nome significa "collegamento furbo") sancisce che ogni drive deve saper fare almeno quattro cose: leggere e scrivere (normalmente per blocchi, ma teoricamente anche uno *stream* di byte), inizializzare il disco fisico preparandolo all'uso, e descrivere il suo stato: occupato o libero, guasto o funzionante, protetto in scrittura o no, contenente un disco o vuoto, ed altro ancora.

SmartPort sa che esistono i dischi SCSI, li riconosce e li tratta in modo adeguato. SmartPort sa distinguere un disco RAM e trattarlo in modo coerente, permettendovi di regolarne l'uso dal pannello di controllo. SmartPort controlla all'accensione se è stato installato nel sistema un disco ROM (un espansione di memoria ROM contenente programmi utilizzabili sotto ProDOS) e permette di usarlo.

SmartPort è insomma una standardizzazione dei device driver di cui avevamo parlato nel resto del capitolo. Esiste in due versioni differenti: la versione standard, creata per Apple IIc, e la versione estesa. La principale differenza è che la versione estesa riconosce tutta la memoria installata su Apple IIgs e permette di leggere e scrivere blocchi di memoria permanente da ogni punto della memoria, mentre la versione standard si limita al blocco zero di memoria (il banco principale).

Sappiamo che i disk drive sono collegati al computer attraverso gli slot oppure i *port* (cioè i pseudo slot incorporati nel com-

puter che possono essere sostituiti agli slot usando il pannello di controllo). Ad ogni slot corrisponde un piccolo spazio in memoria, quello compreso tra FF/Cn00 e FF/CnFF, dove n è il numero dello slot, che contiene il device driver della periferica collegata tramite quello slot.

Possiamo distinguere gli slot connessi ad unità a disco leggendo alcune di quelle locazioni; per le unità a disco infatti...

$Cn01 = $20
$Cn03 = $00
$Cn05 = $03

Inoltre per le unità a dischi da 5" 1/4 Disk II abbiamo che

$Cn07 = $3C

mentre per le unità a disco evolute che usano SmartPort

$Cn07 = $00

Le routine che SmartPort mette a disposizione sono elencate nella tabella che accompagna questo paragrafo. Non tutte sono necessariamente significative per ogni singolo disk drive: anzi, molte appartengono a due gruppi distinti, le chiamate per drive a stream e le chiamate per drive a blocchi. Infine, è possibile assimilare al protocollo di SmartPort altre richieste al drive, specifiche e dipendenti dalla sua natura.

Le due chiamate generiche sono la **status** che restituisce informazioni generali sul disk drive in questione: numero di blocchi contenuti su ogni disco (oppure numero di byte), tipo di disco (RAM, ROM, SCSI, hard eccetera), e qualcosa d'altro; e la **format**, che incarica il disco di dare un formato fisico al disco inserito, preparandolo all'uso.

Tabella: le chiamate di SmartPort		
$00	Status	Tutti i dischi
$01	Read Block	Solo dischi a blocchi
$02	Write block	Solo dischi a blocchi
$03	Format	Tutti i dischi
$04	Control	
$04	Init (reset)	Tutti i dischi
$06	Open	Solo dischi a stream
$07	Close	Solo dischi a stream
$08	Read	Solo dischi a stream
$09	Write	Solo dischi a stream

Ora qualche parola per i programmatori.

SmartPort viene chiamata con un metodo analogo a quello usato con la MLI di ProDOS: l'indirizzo di entrata viene trovato calcolando l'indirizzo del device driver (secondo il metodo di sempre pubblicato nel ProDOS technical manual, cioè nel byte $Cnxx dove xx è il valore del byte $CnFF) e poi aggiungendo tre. Come esempio, ecco la routine in Assembler che espelle un disco da 3" 1/2 dal drive: funziona sia con gli UniDisk che con gli Apple Disk 3.5

```
 LDA SLOTNUM
 CLC
 ADC #$C0                ;Trova numero di pagina
 STA PCH
 STA SMARTCALL+2         ;Salvalo per dopo
 LDX #$0
 STX PCL
 LDY #07                 ;Controlla che sia uno slot con
                         SmartPort
 LDA (PCL),Y             ;SmartPort signature
 BEQ ISSMART             ;Se $Cn07 = 0 è smart
 JMP ERRORE              ;Not smart enough
ISSMART LDY #$FF         ;Trova l'indirizzo del ProDOS De-
                         vice Driver
 LDA (PCL),Y
 CLC
 ADC #$03                ; e ricavane lo SmartPort Address
 STA SMARTCALL+1
 LDA DRIVENUM            ; numero di drive: ammesso da 1 a
                         127 (!)
 STA SMRTUNIT
SMARTCALL JSR $0000      ;Esegue l'operazione
 DFB $04                 ;Parametro per la chiamata di con-
                         trollo
 DW SMRTPARM             ;Puntatore ai parametri
 BCS ERRORE              ;C'è stato un errore, oppure...
 ...                     ; resto del programma
SMRTPARM EQU *
 DFB $03                 ;Param Count, come con MLI
SMRTUNIT DS 1
 DDB $0000               ;CTRLLIST
 DFB $04                 ;Eject opcode
```

8

QUALCOSA DI PIÙ SU PRODOS 16

Dopo avere esaminato, nel precedente capitolo, tutti i sistemi operativi della famiglia Apple II ed avere introdotto anche il ProDOS 16 creato per Phoenix, in questo capitolo diamo qualche notizia in più su quest'ultimo. Qualcuna molto utile, qualcun'altra solo interessante.

8.1 ProDOS 16 e la subdirectory System

Sino a qualche anno fa, prima dell'avvento di Phoenix, l'unico ProDOS esistente era il ProDOS 8.

Il codice macchina che costituisce il sistema operativo ProDOS 8 era contenuto in un file chiamato PRODOS sul disco. In questo modo è possibile destinare un disco a contenere soltanto dati, guadagnando lo spazio normalmente riservato al ProDOS, cancellando quel file. Il disco diviene non lanciabile: se provate a farlo partire ottenete il messaggio "Unable to load ProDOS" (non posso caricare ProDOS).

Con l'avvento di ProDOS 16, le cose sono cambiate. Come abbiamo visto nel capitolo 7, oggi esistono due differenti ProDOS: il ProDOS 8, in grado di funzionare su ogni Apple II; e il ProDOS 16, destinato al solo Apple IIgs.

Notate che ProDOS 16 non sostituisce ProDOS 8 sugli Apple

IIgs, ma gli si affianca: i programmi creati per il solo Phoenix usano ProDOS 16, ma quando scegliete di far funzionare un programma che può operare su tutti gli Apple II, come AppleWorks, ProDOS 16 carica ed esegue il ProDOS 8 che a sua volta carica ed esegue il programma.

Un disco lanciabile creato con ProDOS 16 contiene sempre il file PRODOS e la subdirectory SYSTEM, la quale a sua volta contiene altri file e subdirectory: osservate prego la figura 1.

L'idea generale è che ciascun componente del sistema operativo è contenuto in un proprio file, e viene caricato in memoria quando ce n'è bisogno (qualche volta al lancio del sistema, qualche più tardi). Questo è lo stesso principio fondamentale del sistema operativo di Macintosh, con l'unica variazione che su Mac i componenti sono tutti contenuti in un unico file, chiamato System, che deve venire riscritto interamente ogni volta che si vuole aggiungere o togliere qualcosa, e per di più si rende necessario usare una utility apposita (come la famosa Font DA Mover): su Apple IIgs, più razionalmente, è sufficiente aggiungere quel che vi serve (una fonte di caratteri, un accessorio di scrivania o altro) nell'apposita subdirectory.

Per l'appunto, abbiamo una subdirectory chiamata **fonts** che contiene le immagini dei caratteri grafici; un'altra, chiamata **desk.accs** che conserva gli accessori di scrivania, classici e nuovi.

Nella subdirectory **drivers** vanno i device driver per le stampanti: un *device driver* è un modulo di codice macchina che gestisce una particolare stampante; quando volete fare una stampa, il Print Manager carica l'opportuno device driver – quello adatto alla vostra particolare stampante – ed esegue l'ordine.

La subdirectory **tools** contiene delle routine in linguaggio macchina che vanno ad aggiungersi a quelle contenute nella ROM di Phoenix. Sul toolbox, e sulla distinzione tra tool in memoria permanente e tool in memoria di lavoro, torneremo in un prossimo capitolo.

Infine, la subdirectory **system.setup** contiene eventuali programmi che debbono essere lanciati prima che il computer possa dirsi pronto a funzionare. Un esempio tipico è il programma che nazionalizza il Pannello di Controllo, sostituendo alle scritte in inglese la traduzione italiana (o francese, o hindu).

Restano i file chiamati P8 e P16: questi contengono, l'avrete indovinato, i due sistemi operativi: ProDOS 8 e ProDOS 16.

L'idea di base sta nel fatto che il file PRODOS viene caricato al

lancio del disco, come avveniva sempre: contiene però non un sistema operativo, ma un programma che identifica il tipo di macchina su cui si trova ad operare, quanta memoria è installata, e provvede a caricare quello dei due sistemi operativi che è necessario, insieme a quanto altro serve (accessori, tool, loader...).

Supponiamo che il computer in questione sia un Apple IIgs: viene dunque caricato in memoria ProDOS 16, che assume il controllo della macchina.

ProDOS 16 cerca innanzitutto nella subdirectory SYSTEM alla ricerca di un Program Launcher. Se un tale programma viene trovato, viene eseguito.

Se lo Start non esiste su questo disco, ProDOS 16 cerca sul disco un programma il cui nome termini con .SYS16, come per esempio GsPaint.Sys16, e se lo trova lo esegue.

Se non esiste neppure un programma del genere, ProDOS 16 cede il controllo a ProDOS 8, il quale prende possesso dei 128 Kbyte inferiori della macchina e cerca sul disco un programma il cui nome termina per .SYSTEM, come ad esempio Basic.System oppure AplWorks.System, e passa ad eseguirlo.

Se non si trova neppure un programma del genere, il sistema operativo si blocca visualizzando il messaggio d'errore *** Unable to Find a ".system" program ***

Se ai possessori di Apple IIgs interessa avere dei dischi contenenti esclusivamente programmi ".system" – cioè solo programmi che sono stati creati per tutta la serie Apple II e non per il solo IIgs – il processo di bootstrap (inizializzazione del sistema) può essere notevomente velocizzato usando il solo ProDOS 8.

Su un disco vuoto copiate il file P8, che poi rinominerete PRODOS: avete ottenuto un disco che lancerà immediatamente ProDOS 8. Aggiungete sul disco i programmi System che vi servono e vivete felici.

Se avete dei vecchi dischi basati su ProDOS 8 in versione 1.1.1 o anteriore, fate la stessa operazione: cancellate il file PRODOS e sostituitelo con il file P8 del disco sistema (quello contenente Mouse Desk 2.0 oppure il Finder), che poi dovrete rinominare in PRODOS. In questo modo tutti i vecchi programmi vi permetteranno di usare il Pannello di Controllo, cosa che altrimenti non è possibile.

8.2 ProDOS 16 versioni 1.1 e 2.0

Quando, nel settembre 1986, Apple Computers ha presentato l'Apple IIgs, la macchina era ancora largamente incompiuta. Pare che sia diventato un modo di procedere universalmente diffuso: basti dire che Ibm ha fatto lo stesso di recente con il suo System/2.

Il lancio del computer sul mercato era previsto per la fine di novembre: e gli sviluppatori si sono resi conto che non sarebbero riusciti, entro quella data, a terminare lo sviluppo del nuovo sistema operativo ProDOS 16. Tuttavia, era necessario avere un sistema operativo: senza di quello nessuno avrebbe potuto creare programmi per la nuova macchina.

È stato così creato il ProDOS 16 1.0, che è semplicemente il vecchio ProDOS 8 leggermente modificato in modo da adattarsi a Phoenix e da imitare quelli che saranno i comportamenti del vero, e futuro, ProDOS.

È per questo motivo che i programmi vengono caricati in memoria con quella spaventosa lentezza, è per questo motivo che non è stato messo in circolazione il Finder (un programma tipo MouseDesk, ma veramente all'altezza della macchina).

Il Finder GS, che è stato sviluppato regolarmente, non può funzionare altro che con il vero ProDOS 16: e dunque non poteva venire usato con ProDOS 16 1.0.

Durante l'estate 1987 verrà diffuso il vero ProDOS 16, che porterà il numero di versione 2.0. Oltre ad essere più veloce ed a permettere l'uso di un Finder adeguato, ProDOS 162.0 dispone di molte caratteristiche in più rispetto al suo predecessore d'emergenza. Ad esempio, permette di spostare un file su di un disco da una cartelletta ad un'altra senza doverlo ricopiare fisicamente, ma semplicemente trasportando le informazioni che lo descrivono. Permette alle applicazioni di formattare dei dischi, mentre con ProDOS 8 e con il falso ProDOS 16 1.0 questo non è possibile, e vanno usate delle opportune applicazioni di sistema come MouseDesk, Copy II Plus o le System Utilities GS.

Corollario: non appena sarete in possesso di un disco Apple contenente il ProDOS 16 2.0, copiatelo su tutti i vostri dischetti sostituendo il vecchio ProDOS 16 1.0 ed 1.1, che è solo una revisione minore che permette l'uso del Print Manager). Per sostituire il vecchio ProDOS con quello nuovo, dovete semplicemente

rimpiazzare il file PRODOS ed il file P16 che si trova nella cartella SYSTEM. Potrebbe anche essere necessario aggiornare i file che sono contenuti in System/System.Setup, e forse il ProDOS 8 che si trova nel file P8.

Un consiglio: se possedete almeno due disk drive da 800 Kbyte, oppure uno hard disk (disco rigido), mantenete un dischetto, oppure il disco rigido, sempre aggiornato con il sistema operativo più recente. Ogni qual volta trovate un nuovo tool, una nuova fonte di caratteri che vi piace, un nuovo accessorio di scrivania che vi interessa, installatelo sul disco.

Su quel disco non dovreste conservare nessun programma: solo il sistema operativo ed il Program Launcher – oppure, meglio, il Finder – non appena sarà disponibile – nella sua versione più recente. (Ovvimente, quest'ultima parte del consiglio vale solo per i dischi da 800 Kbyte e non per il disco rigido).

A questo punto, se avrete l'accortezza di usare sempre quel disco per avviare il sistema, sarete certi di usare i programmi al meglio delle possibilità.

La necessità di avere almeno due disk drive è motivata dal fatto che nel secondo drive – nell'unico drive da tre pollici, nel caso dello hard disk – andrà il disco con il particolare programma da eseguire. I possessori di un solo disk drive sarebbero costretti ad un continuo ed infernale scambio di dischi nel drive.

8.3 Il Relocating Loader

Se avete programmato in Basic, lo sapete già: "load" in inglese significa "caricare". Il System Loader di Apple IIgs è un tool del toolbox molto speciale, che lavora fianco a fianco con ProDOS 16 e con il memory manager (il gestore della memoria).

Una applicazione molto spesso deve trattare con i dati su disco; per esempio, un word processor deve leggere e scrivere i suoi documenti, e il Finder deve poter eseguire i programmi.

Una azione concettualmente semplice come il caricamento in memoria del contenuto di un file (un documento memorizzato sul disco) è in realtà composta da più azioni elementari, che potremmo descrivere così:

* Riserva uno spazio adeguato in memoria
* Apri il file

* Leggi il file, un pezzo alla volta, sinché non è letto tutto
* Chiudi il file

Seguono altre operazioni, anche complesse, se il file conteneva l'immagine di un programma da eseguire. In particolare, vanno distinti i casi di una applicazione che deve sostituire quella corrente e di una applicazione che si affianca a quella corrente (come un accessorio di scrivania).

Un primo compito del System Loader è di gestire questo tipo di azioni nel modo più semplice e trasparente possibile. ("Trasparente" è termine che si incontra spesso parlando di computer, e significa "in un modo naturale e logico, che non richieda interventi da parte dell'utente o dell'applicazione che ha richiesto l'azione").

Il System Loader è un attrezzo sufficientemente sofisticato che consente anche operazioni più complesse. Prendiamo il caso di una applicazione che, sommando insieme lo spazio necessario al codice eseguibile ed ai dati, richiederebbe più memoria di quanta sia disponibile dentro il computer.
Non è necessariamente detto che un problema del genere sia insolubile. Se avete provato ad usare AppleWorks, il popolare e potente programma per la famiglia Apple II che comprende un word processor, una base di dati ed uno spreadsheet, avete già incontrato un programma che presenta un caso del genere.
AppleWorks comprende in se qualcosa come 200 Kbyte tra dati e codice, eppure funziona su macchine con soli 128 Kbyte di memoria, lasciandone 56 all'uso dell'utente.
Il trucco sta in una semplice considerazione: AppleWorks realizza, come abbiamo accennato, tre differenti funzioni, che vengono utilizzate una alla volta.
Pertanto, non tutto il codice deve risiedere nella memoria del computer contemporaneamente.
Quello che è necessario fare è distinguere tra la parte del programma che deve essere permanentemente presente nel computer e quella che è necessaria solo in determinate occasioni. Se si presenta un conflitto di memoria, è possibile liberare lo spazio occupato da codice non indispensabile ed assegnare quello spazio ad un pezzo di codice simile che realizza una funzione attualmente richiesta.

Il Loader di Phoenix serve anche a questo: permette di distinguere i componenti di un programma in più parti, alcune delle quali possono essere rimosse dalla memoria quando ve ne sia bi-

sogno. L'applicazione non deve preoccuparsi di gestire questi scambi di spazio, che sono curati dal Memory Manager.

Un altro caso dove il Loader si dimostra indispensabile è quello più significativo, quello che spiega a cosa esso debba il suo nome di Relocating, cioè "rilocatore". Per comprenderlo, facciamo un inciso.

Perché un programma possa venire eseguito ad un computer – o, per essere più precisi, dal suo processore – deve venire codificato in quel linguaggio di programmazione conosciuto come "linguaggio macchina". Dato che la programmazione in linguaggio macchina è inconcepibilmente difficoltosa, i programmatori usano degli strumenti (i linguaggi di altro livello) che consentono di svolgere in maniera più semplice il compito di programmare.

Il programmatore può lavorare, a sua scelta, in Assembler, un linguaggio molto complesso che ha una corrispondenza strettissima con il linguaggio macchina, oppure in un linguaggio più evoluto ma meno efficiente, che sia interpretato (come il Basic) o compilato (come il Pascal e il linguaggio C).

Comunque sia, in qualche modo le istruzioni di questi linguaggi vengono infine tradotte in codice macchina, che il processore può eseguire. Il guaio è che il codice macchina è vincolato ad un punto nella memoria della macchina: cioè, quando viene creato il programma, è necessario scegliere un punto della memoria RAM della macchina in cui il programma verrà caricato ed eseguito, e non sarà possibile farlo eseguire in nessun altro punto della memoria.

Se questa condizione non è troppo vincolante sui computer più semplici (gli Apple IIe e IIc, per esempio, funzionano perfettamente rispettando questo vincolo), quando vogliamo ottenere qualcosa di più sofisticato la condizione diventa troppo restrittiva.

Pensate agli accessori di scrivania: come sarebbe possibile avere tutti gli accessori che vogliamo, se ciascuno fosse vincolato a funzionare in uno ed un solo punto della memoria? Quasi certamente si avrebbe un conflitto di interessi, con l'applicazione e gli accessori che richiedono contemporaneamente le stesse locazioni di memoria.

Per rimediare al problema, si usa un metodo ingegnoso. Tutti i programmi sono creati in modo che possano eseguire venendo caricati nelle primissime locazioni della memoria del computer (dalla numero zero in poi).

Quando poi il programma viene davvero caricato in memoria,

il Loader, che esegue il caricamento, "riloca" il programma. Cioè, va a modificarne tutti i punti chiave in modo che il programma possa funzionare nel punto specifico che gli è stato destinato questa volta. Questo è possibile perché, insieme al codice ed ai dati che formano il programma, viene salvata una tabella che indica quali sono e dove sono quei punti vitali che andranno rilocati. La creazione della tabella è effettuata dall'assemblatore, compilatore o interprete che il programmatore ha usato per creare il programma, senza che il programmatore debba preoccuparsene.

Il System Loader di Phoenix permette, oltre alle tre citate, anche altre funzioni meno rilevanti, come la definizione di un "segmento di inizializzazione", una parte di codice che va eseguita immediatamente non appena caricata in memoria, segmenti assoluti (da caricare in un determinato punto della memoria e non rilocabili) eccetera.

9

SYSTEM MONITOR, IL SIGNORE ASSOLUTO

Anche i primissimi modelli di Apple II avevano qualcosa che li distingueva dai loro rivali ed avversari, un qualcosa in più che faceva la gioia dei programmatori e degli utenti esperti, qualcosa che tutti invidiavano e che pochissimi tentarono, comunque senza successo, di imitare: System Monitor.
In questo capitolo scopriremo cosa sia Monitor ed impareremo ad usare la versione particolarmente potenziata che è disponibile con Apple IIgs.

9.1 Cos'è il Monitor

Monitor è un programma in linguaggio macchina che occupa uno spazio in memoria ROM, e che viene sfruttato da ogni altro programma che risieda nell'Apple.
Il compatto e potente Monitor, questa la sua principale caratteristica, è costituito interamente di subroutine. Per i non programmatori questo termine può essere oscuro: significa, in soldoni, che Monitor mette a disposizione pezzi di programma già pronti per svolgere tutta una serie di compiti frequenti: per esempio accedere alla stampante, fare scritte sul video o spostare informazioni entro la memoria del computer.
Tra le routine a nostra disposizione dobbiamo davvero ricordare la vecchia, fedele COUT1, chiamata anche "fidifo" dall'in-

dirizzo dove si trova $FDF0. COUT1 è quella umile lavoratrice che visualizza ogni carattere sul video, e lavora in coppia con la sua compagna $FDED, che passa il carattere a chi lo richiede: stampante, plotter, disco e chiunque altro.

In monitor stanno anche WAIT, la routine che permette di attendere un tempo fissato, utilizzata specialmente negli effetti sonori; BELL, la routine che suona il campanello quando premete Control–G; HOME, che azzera lo schermo; RDKEY, la routine che legge un tasto; GETLN, la routine che chiama RDKEY per lasciarvi introdurre una riga di caratteri; MOVE e VERIFY, le due gemelline che operano su blocchi di memoria, e tante altre.
Monitor è anche il portiere di casa, il primo a svegliarsi all'accensione del computer, e provvede personalmente a fare un po' di pulizie nella memoria prima di caricare da disco il sistema operativo. Una routine tra quelle attivate è quella chiamata APPLEII, all'indirizzo $FB60.

Monitor è anche un ambiente nel quale possiamo impartire ordini al computer al livello più basso, esaminando la memoria e i programmi, controllando le periferiche e svolgendo altri compiti di cui parleremo tra poco.
Il modo più semplice di accedere al System Monitor è passando dal Basic Applesoft. Accendete la macchina senza nessun disco inserito; dopo poco tempo apparirà la scritta "Check startup device" e la mela che rimbalza sullo schermo. A questo punto, battete Control–RESET (RESET è il tasto che si trova di fianco alla mela colorata in alto a sinistra sulla tastiera, con l'incisione di un triangolo.
Apparirà una parentesi quadra chiusa (oppure una "è" che ne è sinonimo come abbiamo visto nel capitolo 2). Siete in Applesoft Basic senza nessun sistema operativo in controllo della macchina.
Battete CALL –151 <RETURN>. Apparirà il simbolo di "pronto" del Monitor, l'asterisco, fiancheggiato dal cursore.
Per provare una delle routine fornite da Monitor, la APPLEII di cui parlavamo, battete FB60G <RETURN>. Visto?

Molti computer hanno un System Monitor disponibile, ma nessuno eccetto il nostro ne ha uno in ROM. Monitor è la spina nel fianco dei protettori di programmi, che fanno di tutto per impedirvene l'accesso. Con Monitor sempre a disposizione voi potete entrare in ogni programma, per studiarlo e modificarlo. Grazie a System Monitor potete entrare nel cuore della mela, e studiarne i comportamenti e le idiosincrasie dall'interno.

Alcuni dei comandi di Monitor vi sembreranno dapprima traslitterazioni dal babilonese in arabo, e con la stessa utilità di una pistolettata al basso ventre: man mano che diventerete più esperti (e specialmente se prima o poi proverete a programmare Apple IIgs) scoprirete quanto insostituibile sia il Signore della Mela.

9.2 Storia del Monitor

Sono state create cinque differenti versioni del Monitor: quella più recente, creata per Apple IIgs, è di gran lunga la più potente.

La più vecchia viene chiamata **vecchia Rom** ed era usata sui primi Apple II, per intenderci quelli che non avevano neppure Applesoft ma usavano invece l'Integer Basic. All'accensione non veniva inizializzato il Basic, né lanciato il disco come avviene in tutti i modelli successivi; inoltre, il RESET riportava sempre e inesorabilmente in Monitor (per questo motivo molti "hackers", quei simpatici giovanotti che si divertono a sproteggere e modificare i programmi altrui, si procurano un Apple IIe, una ROM con il vecchio monitor e la installano dentro il computer. Poi interrompono il programma "protetto" nel punto scelto e scavano, scavano, scavano...).
Il vecchio Monitor aveva anche il MiniAssembler, un assemblatore (cioè un aiuto per sviluppare programmi in linguaggio macchina) operante su una istruzione la volta, molto utile per i primi approcci al linguaggio macchina. È stato tolto nelle versioni successive e reintrodotto nelle ultime, come altri due comandi: STEP e TRACE, usati nel debugging dei programmi in linguaggio macchina ed utilissimi a quello scopo.

Con l'avvento di Applesoft e dell'Apple II+ il Monitor venne modificato; vennero introdotti i vettori di RESET (ora il Control-RESET può essere programmato ad eseguire una routine creata dal programmatore... dovrebbe servire a far ripartire programmi bloccati). Monitor venne decurtato dei comandi step e trace per far posto alla caratteristica chiamata **autostart**: come sapete, ora Apple alla accensione lancia il disco con il sistema operativo.

Una terza versione si è avuta con l'introduzione di Apple IIc: sono state tolte le routine relative alla gestione delle cassette, e

quindi i comandi R e W di Monitor, per aggiungere all'assemblatore i nuovi op-code. Come abbiamo detto, infatti, il microprocessore 65C02 di Apple IIc possiede 10 istruzioni e 2 modi di indirizzamento in più del 6502 usato sui modelli precedenti; il Monitor del IIc riconosce questi nuovi codici e li disassembla regolarmente.

Con i modelli più recenti sono state introdotte altre modifiche sostanziali al RESET. Dal IIe in poi, è necessario premere Control insieme a RESET per reinizializzare il sistema a caldo – eliminando così il "problema Vinzo", così battezzato dal sottoscritto in disonore di un mio amico, tale Andrea Vinzoni, che una volta premette RESET per errore durante una partita contro il computer che durava da tre ore.

Con Control-Mela Vuota-RESET il sistema viene rilanciato da freddo, come se venisse spento e riacceso. Con Control-Option-Reset è possibile impostare il computer per funzionare indifferentemente nei paesi con corrente elettrica a 50 hertz (secondo il sistema europeo) oppure 60 hertz, secondo il sistema americano. Con Control-Mela Piena-Option-Reset (un ordine che richiede notevole destrezza manuale) il computer esegue una serie di test di autodiagnosi. Dopo qualche minuto se ne esce con la scritta "System ok." per segnalare di essere in salute, oppure indica un codice di errore.
I più recenti Apple IIe ed Apple IIc hanno ricevuto il beneficio di una revisione che ha potenziato il Monitor (ne parliamo nella appendice A). Per Apple IIgs si è giunti alla quinta riscrittura di Monitor, che ha raddoppiato il numero di comandi disponibili. Nel prossimo paragrafo vedremo quali e quanti comandi sono a nostra disposizione, da battere dopo quell'asterisco tentatore.

9.3 Il nocciolo della mela

Tutti i comandi di Monitor sono composti da un solo carattere, eventualmente accompagnato da parametri. Per esempio, provate a battere:

Control–^ ?

cioè prima insieme "control" e "^", poi il punto di domanda (parametro).

I numeri sono rappresentati in esadecimale, il sistema numerico in base 16 che usa oltre alle dieci cifre sei lettere dalla A alla F. I primi numeri interi, rappresentati in esadecimale, sono 0 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12...

Spesso un parametro è l'indirizzo di una locazione di memoria; in quel caso viene indicato con il numero di banco, la sbarra, e poi il numero di pagina e di byte scritti di seguito. La prima locazione di memoria è la 00/0000 e l'ultima è la FF/FFFF

Nel paragrafo precedente abbiamo usato il comando FB60G: G è uno dei comandi di Monitor, (sta per "Go", cioè "Vai!"), ed è l'equivalente del RUN Basic per programmi in linguaggio macchina; dunque il nostro comando ha dato a System Monitor un indirizzo da cui partire ed il comando di eseguire la routine, un po' come quando comandiamo in Basic RUN 200. Un suo fratello maggiore introdotto in Apple IIgs per la prima volta, X (sigla di "eXecute") permette di eseguire programmi ovunque nella memoria e non solo in quella disponibile anche ai vecchi Apple II: si scrive l'indirizzo completo del numero di banco, come in E1/0000X. Ci si aspetta che le routine eseguite siano delle subroutine che terminano con RTS o RTL rispettivamente, e quando il codice di ritorno viene eseguito il controllo torna al Monitor. Per eseguire un programma senza che il controllo torni a Monitor, entrandovi dunque con un JMP, si usa il comando R (run).

Altri semplici comandi di Monitor servono per leggere una locazione di memoria (come il PEEK di Applesoft) o per scriverci (come POKE).

Per leggere il contenuto di una locazione, basta scrivere il suo indirizzo e battere RETURN. Per esempio, provate con FF/FBB3 <RETURN>; il computer risponde con

FF/FBB3: 06 – .

comunicandovi il valore contenuto in quella locazione (06). Al valore viene fatto seguire il suo equivalente ASCII, se esiste (abbiamo parlato del codice ASCII nel paragrafo 2 del capitolo 2).

Per cambiare i contenuti di un byte dovete battere il numero della locazione scelta, due punti, il valore sostituendo. Se aggiungete altri numeri separati da spazi questi verranno posti nelle locazioni successive alla prima indicata; è così che si procede per introdurre i programmi in linguaggio macchina copiati dalle riviste. Per esempio, battete 300: 20 3A FF 60 <RETURN> e poi

300G <RETURN>: avete appena introdotto ed eseguito un programma in linguaggio macchina, che suona il campanello. Si può anche introdurre una stringa di caratteri codificata in modo ASCII, usando l'operatore " (doppie virgolette) come in

01/2000: "MisterAkko"

Se usate invece delle virgolette l'apice la stringa viene inserita capovolta in memoria.

Veniamo al comando più importante. Battete FF/D000L.
Il comando L sta per "list", ed attiva il disassemblatore incorporato in Monitor. Questo simpatico e preziosissimo collaboratore ci fornisce per ciascun valore trovato in memoria l'operazione macchina connessa; per essere più precisi, a ciascun opcode fa corrispondere la label mnemonica dell'istruzione. Ottima cosa, ideale per leggere i programmi altrui, decodificarne le routine e magari togliere quelle noiose e patetiche "protezioni"...

I comandi M (move) e V (verify) hanno la stessa sintassi: (destinazione)<(inizio).(fine)M per move – e V anziché M per verify. Se battete 4000<2000.3FFFM trasferirete l'immagine contenuta in HGR1 in HGR2 (l'originale resterà immutato). Verify compara i due blocchi di memoria e vi segnala eventuali differenze. Per riempire una zona di memoria con un valore usiamo Z, scrivendo (valore da usare)<(locazione iniziale).(locazione finale)Z.
C'è bisogno di dire che l'uso indiscriminato dei comandi M e Z può distruggere dati e programmi eventualmente in memoria? Spero di no.

Con il ! (punto esclamativo) invochiamo il Mini Assembler, utile ai programmatori in linguaggio macchina per provare le loro routine. Con il mini macchina, pronto per essere eseguito.
Per esempio, provate con...

```
11<00/2000.00/3FFFZ
!300: JSR $FF3A
 : JSR $F3E4
 : RTS
```

poi battete un paio di volte RETURN da solo. Adesso eseguite il programma che avete battuto, comandando 300G. Carino, no?

Adesso però siete rimasti bloccati sullo schermo grafico di alta

risoluzione, che vi occupa quattro quinti dello schermo. Potete ritornare nel modo testo battendo Control–T e poi RETURN.

Ecco un altro esempio interessante:

```
1=e
!900: STZ FE
 : LDA #20
 : STA FF
 : LDY #0
 : LDA (FE),Y
 : JSR FDED
 : INC FE
 : BNE 913
 : INC FF
 : CMP #0
 : BNE 908
 : RTS
```

A questo punto battete due volte RETURN a vuoto. Poi...

```
900G
908G
```

Altri comandi di monitor sono I ed N (inverse e normal); (inizio).(fine) per vedere i contenuti di più locazioni di memoria – e si può terminarlo anticipatamente premendo Control–X; CONTROL–P; analogo al PR# di Applesoft (1 CTRL–P attiva la stampante); CONTROL–K, analogo al IN# di Applesoft; CONTROL–C per tornare ad Applesoft con il programma Basic intatto e CONTROL–B per tornarvi senza di esso. Con il comando Q (quit) lasciate il Monitor e rientrate nel programma che stavate usando: può servire se Monitor è stato invocato da una applicazione o da un accessorio di scrivania.

I programmatori Assembler troveranno utilissime le eccellenti capacità di Monitor per eseguire calcoli aritmetici in esadecimale: per esempio potete eseguire una somma chiedendo

```
567DA4+1165470A
```

o risolvere similmente una sottrazione (con –), una moltiplicazione (con *) o una divisione (con _, la sottolineatura, perché il carattere / è già usato per altre operazioni).

È possibile far eseguire al Monitor le conversioni di sistema

numerico da decimale ad esadecimale e viceversa. Per esempio, scrivendo

```
FB60=
```

Monitor vi risponde che quel valore esadecimale in decimale viene scritto come 64352. Se battete

```
=8192
```

System Monitor traduce il numero decimale in esadecimale.

Monitor può venire usato per ricercare un valore (numerico o alfabetico) nella memoria. Per esempio, per cercare se nel banco 1 di memoria si trova il nome "MisterAkko" scriveremo

```
ç"MisterAkko"ç<01/0000.01/FFFFP
```

la lettera P finale è il comando vero e proprio e sta per Pattern search. I valori da cercare vanno inclusi tra le sbarre inverse e non normali: per ottenere quel carattere bisogna premere il tasto con la c francese, come abbiamo scoperto nel capitolo 2.

Battendo Control–E e poi RETURN il computer visualizza il valore attuale dei registri del microprocessore e degli pseudoregistri. Sono nell'ordine...

A accumulatore 65816
X,Y registri indici 65816
S stack pointer 65816
D registro diretto 65816
P stato del processore 65816 (flags)
B banco prescelto (DBR, data bank register 65816)
K banco di programma (PBR, program bank register 65816)

M stato della macchina
Q pseudo registro Quagmire
L banco Language card selezionato
m modo dell'accumulatore (0= 16 bit, 1=8 bit)
x modo dei registri (0=16 bit, 1=8 bit)
e modo emulazione (0=nativo, 1=emulazione)

Lo status degli pseudo registri m, x, e ha effetto sul modo in cui sono interpretati i comandi L, G ed R. Il valore dei registri e pseudoregistri può venire cambiato da Monitor battendo il nuovo valore, il segno di uguale, e il nome del registro (in questo ca-

so e solo in questo è importante distinguere le minuscole dalle maiuscole in un comando a Monitor).

Per esempio, per entrare in modo emulazione Apple IIc si scrive:

1=e

Oltre agli pseudo registri sin qui elencati se ne aggiunge un altro non visualizzato con Control–E: lo pseudo registro F (il Filtro). Serve a filtrare i valori ASCII inseriti con le virgolette in memoria: il valore del filtro viene ANDed alle lettere ed è significativo per scegliere il formato ASCII con bit alto settato o resettato.

Infine, lo speciale comando =T permette, da System Monitor, di conoscere la data e l'ora che vengono lette dall'orologio di Apple IIgs.

10

PER PROGRAMMARE

Questo capitolo è dedicato a tutti quelli che vogliono imparare i principi della programmazione dell'Apple IIgs, che sia in Basic o in Assembler. Non pretendo di esaurire l'argomento, ma di introdurlo.

10.1 La mappa di memoria

La memoria dell'Apple IIgs, come quella dei precedenti modelli, è divisa in banchi, pagine e byte. Provate ad immaginarla come una enciclopedia composta di tanti volumi (i banchi): in ciascun volume ci sono 256 pagine, e ciascuna pagina è composta da 256 parole, i byte.

La quantità di banchi nella memoria dipende da voi: ogni banco contiene, ovviamente, 256 * 256 byte, cioè 64 Kbyte e dunque un Apple IIgs che viene venduto con 256 Kbyte di RAM e 128 Kbyte di ROM parte con 6 banchi. La capacità teorica massima è di 256 banchi, quella pratica imposta dal sistema operativo è di 146 banchi (128 di RAM, 16 di ROM, 2 di slow RAM), il che significa che c'è comunque molto spazio per la crescita.

Un Apple IIgs in versione base ha installati i banchi $00 e $01 (i 128 Kbyte dell'Apple IIc), i banchi $E0 ed $E1 (i banchi di display), $FE ed $FF (la ROM). Quando vengono inserite espansioni di memoria la RAM aggiunta viene vista come banco $02, poi

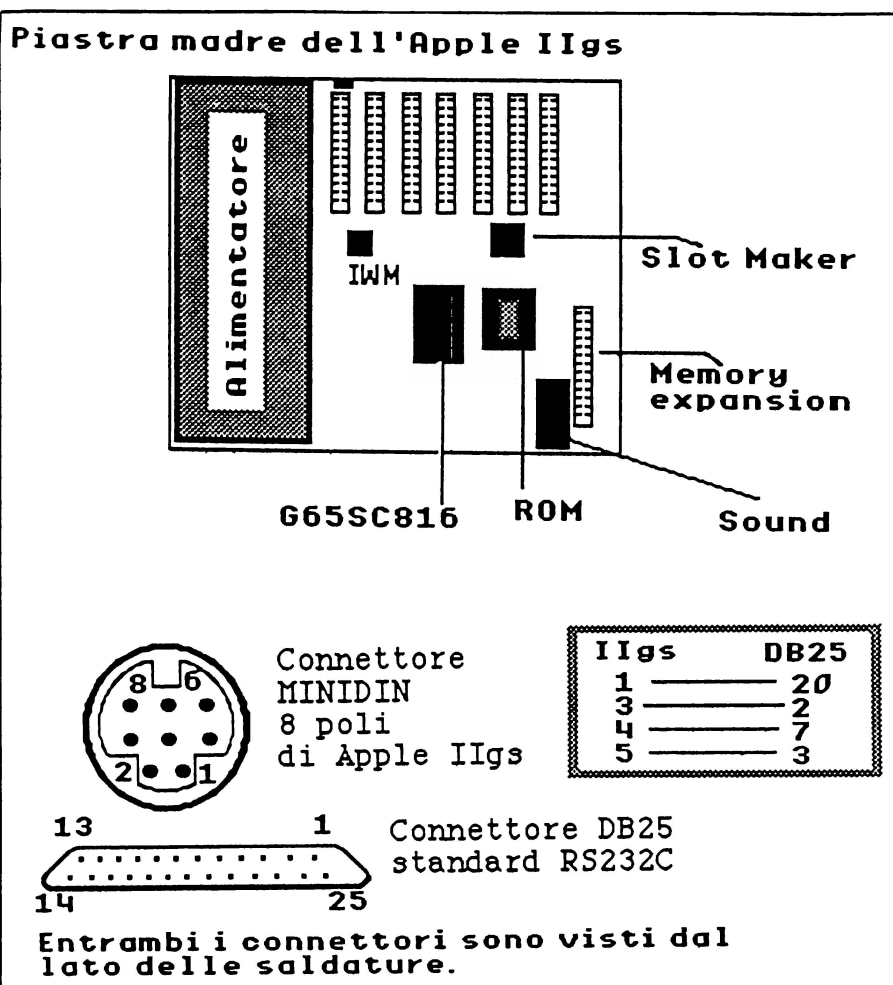
$03 e così via sino a $7F. Se e quando viene aggiunta della ROM, invece, decresce il numero di banco: $FD, $FC e così via sino a $F0.

Ogni banco può contenere parte di un programma, dati o valori fasulli ed inutili; all'accensione tutta la memoria RAM contiene "immondizia", ed è la ROM, la memoria permanente, a predisporla nella maniera esatta.

Nella tabella qui di seguito trovate una mappa della memoria principale di Apple IIgs, quando opera in modo nativo e quando opera in modo emulazione (nel secondo caso la mappa è identica a quella di un Apple IIc). I numeri sono normalmente espressi secondo il sistema esadecimale (indicato dal simbolo "$"). Il sistema esadecimale è il sistema numerico usato internamente dalla Mela, e gli indirizzi di memoria sono ricordati più facilmente in esadecimale, una volta che ci fate l'abitudine: infatti 256 in esadecimale diventa $100.

I banchi $E0 ed $E1 contengono l'equivalente per Apple IIgs di quello che per i precedenti modelli erano le aree di display grafico o testo, e l'area di input/output, la cosiddetta **pagina hardware** $00C0xx.
Per esempio, i caratteri letti dalla tastiera in un Apple IIe sono disponibili al processore nella locazione $C000 (del banco zero), mentre in Phoenix la funzione equivalente avviene in $E0C000.
Perchè un programma creato per i vecchi modelli possa funzionare anche su Phoenix è necessario che gli accessi nel banco zero e nel banco uno vengano riportati anche nei banchi $E0 ed $E1: si chiama **shadowing** ed è uno dei compiti istituzionali di FPI (si veda anche il paragrafo 3.3).
Le locazioni delle aree di display sono immutate rispetto ai modelli precedenti ma trasportati in quei banchi: per cui, per esempio, nell'Apple IIc la alta risoluzione HGR pagina uno sta tra gli indirizzi $002000 e $003FFF mentre nell'Apple IIgs va da $E02000 a $E03FFF.
La super alta risoluzione introdotta per la prima volta nell'Apple IIgs occupa lo spazio tra $E14000 e $E1BFFF e non viene normalmente shadowed, anche se è possibile ottenere questo effetto. Si noti che la super alta risoluzione è supportata dal Toolbox e quindi non è necessario conoscere il suo indirizzo fisico: per disegnare figure determinare basta fare una richiesta a QuickDraw II.
Esiste solo una pagina di super alta risoluzione, mentre tutti gli altri modi grafici hanno due pagine, il che permette di effettuare

animazioni più pulite (si veda il mio articolo "Cosa c'è dietro Conan" in **Bit** n° 77 del novembre 1986). Nell'Apple IIgs le animazioni in super alta risoluzione vanno effettuate tutte in velocità, usando le istruzioni di memory move del 65816, che purtroppo non sono molto efficienti: andranno sviluppate tecniche di sincronizzazione col pennello video, magari usando gli interrupt collegati a VBLINT.

L'uso degli schermi grafici nella memoria dei banchi $E0 ed $E1 limita il loro uso per altri scopi (cioè per contenere un programma). Un Apple IIgs ha un minimo di 256 Kbyte di memoria RAM, ma di questi solo 168 Kbyte sono disponibili per i programmi. Ecco un buon motivo in più per comprare una espansione di memoria come abbiamo anticipato nel capitolo 1.

10.2 Chiamate a System Monitor

Caveat emptor! (Not For Mere Mortals, ovvero Do Not Feed Assembley Language Programmers). Paragrafo solo per programmatori Assembler.

I progettisti di Phoenix hanno reso standard ed ufficiali molti punti d'ingresso al System Monitor, riferentisi alle routine più comunemente usate dai programmatori Assembler. L'elenco completo, comprendente le note tecniche su ogni routine ed i parametri necessari è riportato nel **Firmware Manual.** Nella tabella che accompagna questo paragrafo tròvate semplicemente nomi ed indirizzi delle routine confermate, giusto il minimo nel caso sappiate già di cosa stiamo parlando.

Mentre i manuali tecnici dei precedenti modelli elencavano giusto giusto quattro routine come entry point ufficiali, questa volta alla Apple hanno fatto il contrario, adottando anche punti di ingresso perfettamente disusati (alzi la mano chi dal 1979 in poi ha chiamato in un programma Assembly la routine TOSUB). La mia tabella indica solo i punti di ingresso effettivamente usati (ho usato come riferimento i cookbook di Don Lancaster).

$F800	PLOT	$SB93	SETTXT
$F80E	PLOT1	$FB40	SETGR
$F819	HLINE	$FB4B	SETWND
$F828	VLINE	$FB5B	TABV
$F832	CLRSCR	$FB60	APPLEII
$F836	CLRTOP	$FB6F	SETPWRC
$F847	GBASCALC	$FBC1	BASCALC
$F85F	NXTCOL	$FBD9	BELL1
$F864	SETCOL	$FBE4	BELL2
$F871	SCRN	$FBF4	ADVANCE
$F88E	INSDS2	$FBFD	VIDOUT
$F8D0	INSTDSP	$FC10	BS
$F940	PRNTYX	$FC1A	UP
$F941	PRNTAX	$FC22	VTAB
$F944	PRNTX	$FC24	VTABZ
$F948	PR3BLNK	$FC42	CLREOP
$F94A	PRBL2	$FC58	HOME
$FAD7	REGDSP	$FC62	CR
$FB1E	PREAD	$FC66	LF
$FB2F	INIT	$FC70	SCROILL

$FC9C	CLREOL	$FE84	SETNORM
$FCA8	WAIT	$FE89	SETKDB
$FDOC	RDKEY	$FE8B	INPORT
$FD35	RDCHAR	$FE93	SETVID
$FD67	GETLNZ	$FE95	OUTPORT
$FD6A	GETLN	$FF2D	PRERR
$FD6F	GETLN1	$FF3A	BELL
$FD88	CROUT1	$FFSF	RESTORE
$FD8E	CROUT	$FF4A	SAVE
$FDDA	PRBYTE	$FF58	IORTS
$FDE3	PRHEX	$FF59	OLDRST
$FDED	COUT	$FF65	MON
$FDFO	COUT1	$FF69	MONZ
$FE2C	MOVE	$FFA7	GETNUM
$FE80	SETINV		

L'indirizzo più importante **non** convalidato è quello $FE61 (LIST) per usare il Disassembler. Notate che tutte le routine di input/output si comportano come le loro più vecchie sorelle (a differenza di quanto avveniva nel IIc dove erano state potenziate per lavorare anche in modo 80 colonne) e quindi si può creare qualche problema nell'usarle con 80 colonne. Ad esempio, VTABZ non serve: allo stesso scopo va chiamata TABV.

È interessante notare che nel Toolbox esiste un un insieme di routine, chiamate i TextTools, create esplicitamente per aiutare l'uso del display di testo: queste rendono più semplice l'accesso alle routine BASCALC, VTABZ e le loro sorelle.

Quando Phoenix agisce in modo simulazione, viene usata come ROM la zona di memoria tra $FF/C100 e $FF/FFFF, che contiene Applesoft Basic e i tradizionali punti d'ingresso del Moitor. Alle periferiche collegate via slot è di nuovo possibile usare la zona di memoria da $C800 a $CFFF per le routine di input output, purché la ROM sulla scheda venga disinserita dopo ogni chiamata, esattamente come avveniva sull'Apple IIe.

Applesoft Basic è sempre il solito, "plain old vanilla fudge". Gli unici ritocchi minimi sono gli stessi del IIc: il parser accetta comandi in minuscolo, il comando LIST funziona anche in 80 colonne. I comandi per registratore a cassette (SHLOAD, SAVE, LOAD, STORE e RECALL) sono disinnescati e se richiamati puntano al command handler di Ampersand (&).

10.3 Il Toolbox ed i suoi punti di ingresso

Toolbox (scatola di attrezzi) è il nome con il quale vengono chiamate convenzionalmente le routine della ROM di Apple IIgs e le corrispondenti del fratellone Macintosh. Il concetto di toolbox è un concetto nuovo, introdotto nei personal computer della seconda enerazione, e piuttosto brillante: anziché lasciar reinventare la ruota a ciascun programmatore, che normalmente deve scrivere ogni volta pezzi di programma anche per i compiti più umili e comuni, come pulire lo schermo, si preferisceFar eseguire tutti questi compiti ad un programmatore eccellente: il codice risultante viene conservato nelle memorie ROM, e tutti i programmi possono usarlo seguendo il protocollo fissato.

Le routine sono raggruppate a secondo delle loro funzioni: ogni gruppo di routine è chiamato un tool. E possibile aggiungere dei tool a quelli presenti in ROM caricandoli in memoria RAM, ed è possibile sostituire i tool in ROM con nuove versioni RAM, usando un.. tool, chiamato Tool Locator!

Il protocollo fissato per chiamare il Toolbox è simile sia a quello usato nel Macintosh sia, (forse un po' meno) a quello in uso già da tempo con il ProDOS. Vediamolo dal punto di vista del programmatore assembler: resta inteso che un programmatore Pascal, C o Basic si limita ad usare le procedure e funzioni messe a disposizioni dal linguaggio, che pensa poi ad interrogare il toolbox secondo le convenzioni.

Si carica nell'accumulatore una word (due byte) con il byte basso che indica il tool voluto (per esempio, $0E indica il Window Manager, oppure $15 indica il Dialog Manager) e il byte alto indica la routine (per esempio, dentro il Window Manager $1B indica GrowWindow, la routine che permette di modificare le dimensioni della window, oppure $0F identifica SetFrameColor che consente di cambiare il colore della cornice). Alle routine vanno spesso comunicati dei parametri (per esempio con SetFrameColor va indicato che colore si vuole usare) e qualche volta si ottengono valori in restituzione (per esempio Get Time, che interroga l'orologio per conoscere data ed ora, deve passare il risultato al chiamante). Il passaggio dei parametri è fatto via stack, sia in input che in output: particolare attenzione va dunque dedicata all'ordine in cui i parametri vengono passati ed alla loro dimensione (se si passa un byte quando serviva una word sono

guai seri, perché lo stack viene alterato e il sistema può bloccarsi. Una volta sistemati i parametri, è sufficiente effettuare una chiamata a subroutine JSL (jump subroutine long) all'indirizzo del **dispatcher,** la routine che si occuperà di far eseguire la chiamata dalla routine del toolbox indicata. Passando per il dispatcher si evitano inconvenienti nel caso in cui le routine vengano modificate, modificando le loro dimensioni e quindi i punti di ingresso. L'indirizzo del dispatcher è $E10000: non che memorizzarlo serva a molto, perché l'accesso è mascherato nei linguaggi evoluti, ed in Assembley vengono fornite delle macroistruzioni per le chiamate.

I tool contenuti nel toolbox sono i seguenti:

00: Non esiste un tool numero zero. Quando si trova un riferimento al tool zero, va interpretato come riferimento al sistema operativo, ProDOS 16.

01: Tool Locator: consente l'aggiunta di tool al toolbox.

02: Memory Manager: gestisce la memoria RAM del computer concedendone l'uso a chi lo richiede.

03: Miscellaneous Tools: tutto quello che non rientra nelle altre voci, come l'uso dell'orologio, della RAM permanente alimentata dalla batteria, degli slot eccetera.

04: QuickDraw II: routine grafiche evolute per disegnare cerchi, ovali, quadrati e rettangoli pieni e vuoti; archi, segmenti, scritte in ogni stile, eccetera.

05: Desk Manager: gestore delle icone e dello schermo video, permette di usare gli accessori di scrivania.

06: Event Manager: permette ad una applicazione di controllare e reagire appropriatamente alle azioni dell'utente, come i movimenti del mouse e l'uso della tastiera.

07: Scheduler: risolve i problemi di condivisione del computer. Quando un programma (per esempio un accessorio) richiede l'uso di una risorsa (per esempio la stampante) che è già in uso, si rivolge allo Scheduler che gli riserverà la risorsa non appena disponibile.

08: Sound Manager: uso di basso livello del generatore di suoni.

09: Apple DeskTop Bus Tools: gestione della tastiera, del mouse e degli altri eventuali strumenti di input.

10: SANE: routine matematiche di alta precisione (vedi il prossimo paragrafo).

11: Integer Math Tools: routine matematiche semplici, di matematica intera, e routine di conversione di formato.

12: Text Tools: gestione dello schermo di testo (inteso come alternativo allo schemo grafico).

13: riservato per future modifiche.

14: Window Manager: gestione – creazione e manipolazione – delle finestre.

15: Menu Manager: gestione dei menù a discesa e delle voci che contengono.

16: Control Manager: gestione dei controlli, cioè dei bottoni, sfogliatori, indicatori, ed altri oggetti manipolati con il mouse.

17: Relocating Loader: lavora in concerto con ProDOS 16 per caricare le applicazioni in memoria ed eseguirle.

18: Auxiliary QuickDraw: una espansione a QuickDraw II che contiene altre utilità grafiche, che complementano quelle standard.

19: Printer Manager: gestione della stampante e delle comunicazioni. Consente un semplice ma sofisticato uso della stampante laser.

20: LineEdit: routine per creare e modificare scritte sullo schermo, usando il mouse per taglia e incolla, selezione del testo, eccetera.

21: Dialog Manager: gestione delle finestre di colloquio con l'utente basate sui controlli del Control Manager, e delle finestre dove appaiono gli avvisi e i messaggi d'errore.

22: Scrap Manager: gestione dei taglia e incolla tra programmi differenti e dell'archivio appunti.

23: Standard File Operations: alcune routine già pronte che permettono di far scegliere all'utente un disco e un file, usando il mouse.

24: Disk Utilities: questo tool non è descritto in nessun manuale della Apple Computers, dunque non posso sapere quale sia il suo campo d'azione.

25: Note synthesizer: uso di alto livello del generatore di suoni (descritto nel prossimo capitolo).

26: Note sequencer: uso di alto livello del generatore di suoni (descritto nel prossimo capitolo).

27: Font Manager: tratta gli stili dei caratteri di stampa, permette di sottolineare, di mettere il grassetto, di ingrandire e rimpicciolire le scritte, eccetera.

28: List Manager: un aiuto ai linguaggi di programmazione ed alle applicazioni sofisticate, consente di creare e trattare le liste di variabili dinamiche.

Ovviamente, ai tool si aggiunge il sistema operativo, ProDOS, che quando si affianca al toolbox si comporta come un tool, e si mischia con esso per talune prestazioni particolari come il display della finestra che mostra i file utilizzabili sul disco e permette di sceglierne uno con il mouse.

Il toolbox dell'Apple IIgs è stato reso il più possibile simile a quello di Macintosh, in modo da permettere la scrittura contemporanea di programmi per le due macchine con pochissime modifiche tra l'una e l'altra versione. Ovviamente esistono differenze tra i due toolbox: ad esempio non esistono nel Mac i Text Tools (perché Macintosh non ha un display di solo testo) o il Note Synthetizer (perché il generatore di suoni del Mac è molto meno sofisticato e potente del chip Ensoniq di Phoenix); d'altra parte Apple IIgs manca del Resource Tool, creato per facilitare la creazione di versioni nazionali dei programmi Macintosh. Anche all'interno dei tool che hanno riscontro nelle due macchine esistono diversità: il QuickDraw di Mac è più completo e molto più veloce di quello di Phoenix, mentre il Window Manager di Phoenix è altrettanto migliore di quello del Mac.

10.4 Uso di SANE

Merita una menzione particolare SANE, la libreria di funzioni matematiche di altissima precisione che costituisce uno dei tool del toolbox.

SANE (Standard Apple Numeric Environment) è stata pensata alla Apple nel 1983, seguendo i principi di standardizzazione del

protocollo IEEE 754: un insieme di convenzioni e di principi ai quali avrebbero dovuto attenersi le routine matematiche secondo i principi più recenti di software engeneering. Nello sviluppare il Floating Point Package per Apple Macintosh i programmatori si sono attenuti alle specifiche SANE, ed ora la libreria è a disposizione anche nelle ROM dell'Apple II (era già stata usata anche per i modelli precedenti, ad esempio in Instant Pascal, ma ovviamente risiedendo in memoria RAM).

Per citare i manuali Apple, l'esigenza di creare una libreria standard di funzioni matematiche è dovuta al fatto che troppo spesso per un computer A + B può essere diverso da B + A, o che anche se A è diverso da B, eseguendo la sottrazione A − B otterreste zero.

SANE fornisce un ambiente numerico consistente e preciso, e permette al programmatore di rilevare le **eccezioni** (underflow, overflow, divisione per zero ed altri), di adeguare l'ambiente di calcolo alle sue esigenze (per esempio specificando in che direzione vanno eseguiti gli arrotondamenti). Ovviamente, sono fornite funzioni e procedure da richiamare, ovvero SANE si comporta in tutto e per tutto come un altro tool del toolbox.

Tra le funzioni fornite si trovano routine di conversione, di aritmetica intera e reale, di trigonometria e di matematica finanziaria, e una funzione per ottenere numeri pseudo casuali.

10.5 Programmazione del 65C816

William Mensh è stato uno dei progettisti del 6502 alla MOS Technology, ed in seguito è passato a fondare, dirigere e lavorare per una nuova compagnia, la Western Design Center Inc.

Il processore 65C816 è il suo ultimo parto: il processore a sedici bit nella famosissima e fortunatissima famiglia 65xx, adottato nell'Apple IIgs.

Per un esame, partiamo da quanto rivela il nome: le prime due cifre rivelano la famiglia di appartenenza. Il 65C816 possiede tutti gli opcode del vecchio 6502 e del suo successore il 65C02 usato nei più recenti Apple II e BBC. La lettera C, poi, ci rivela che si tratta di un microprocessore a basso consumo di energia, in tecnologia CMOS, dunque adatto per computer portatili e com-

patti. Infine le ultime tre cifre svelano il suo più importante segreto.

Il 65C816, infatti, può operare esattamente come un 65C02, in "modo emulazione"; in realtà, una volta accesolo, si comporta esattamente come un 65C02 a 8 bit, e non vi è modo di distinguerlo. Un nuovo operatore, però, è stati aggiunto: eseguendolo, il 65C816 rivela la sua natura di 16 bit puro (questo significa, tra parentesi, che NON è compatibile pin per pin col 65C02 e dunque non è possibile sostituirlo al vecchio processore nei computer già esistenti).

Al bit di emulazione fanno fianco due altri bit, chiamati x ed m: indicano rispettivamente la lunghezza di modo nativo (16 bit) o emulazione (8 bit) dei registri indice dell'accumulatore. Tutti e tre i bit indicano l'emulazione col valore 1. In modo nativo con accumulatore ad otto bit si dispone di un secondo accumulatore, chiamato B.

Anche nella sua nuova veste, il processore mantiene inalterati opcode e modi di indirizzamento tipici dei fratelli minori. I registri sono gli stessi (accumulatore, X, Y, program counter, status) del 65C02, ma passano tutti a sedici bit, eccetto il program counter che è capace di 24 bit, giungendo così ad indirizzare un massimo di sedici megabyte, come il Motorola 68000. Si aggiunge un nuovo registro, chiamato "registro diretto", a sedici bit, usato per i nuovi modi di indirizzamento. Lo stack non è più limitato a soli 256 byte, la maggior limitazione del 6502, ma può raggiungere 64K.

Nel registro di status si aggiunge un bit che segnala se il processore sta operando in modo sedici oppure otto bit.

C'è un dettaglio che va tenuto in gran considerazione: il program counter resta normalmente bloccato in un banco di 64 Kbyte, e non passa al successivo (neppure se una istruzione scavalca il banco). Per scrivere programmi più lunghi di 64 Kbyte è sufficiente usare un JMP lungo. Non ci sono limitazioni simili per i dati: il Data Bank Register resta incrementato correttamente.

Veniamo alle nuove operazioni e modi di indirizzamento: tra le prime, notiamo tutte le operazioni più sofisticate che programmatore di linguaggio macchina possa chiedere, con la unica notevole eccezione delle operazioni di divisione e moltiplicazione. Vi sono invece operazioni per spostare blocchi di byte in memoria, il branch lungo (sino a 32K di distanza dal punto del salto, anziché solo 127 byte come nel 6502), ed i JMP, JSR lunghi ovunque nei 16 megabyte di spettro. Un nuovo modo di inter-

rupt per l'inizializzazione del sistema da software è introdotto con l'istruzione STP, e vi sono istruzioni come WAI per attendere un interrupt, la test and set e la test and reset che invitano allo sviluppo di sistemi operativi in multiprogrammazione.

Con il 65C816 si ha un totale di 256 operazioni possibili, coprendo così tutti i codici macchina che il byte può offrire; una istruzione particolare, chiamata WDM, è riservata a futuri sviluppi ed attualmente non compie nessuna operazione: in futuro un microprocessore anora più potente potrà emulare il 65C816 e poi, svegliato con la WDM, procedere ad ancora ulteriori mirabilie.

Per ultimo, un gioellino; l'operatore COP, che permette l'uso di un coprocessore matematico se installato nel sistema, come per lo 8086 del PC IBM.

Il vero punto di foza del 65C816 sono comunque i modi di indirizzamento dei dati, tanto numerosi e sofisticati da non avere pari in nessun microprocessore che il sottoscritto conosca. Sono, in totale, ventiquattro, e se qualcuno dei miei lettori ha provato a cimentarsi con il linguaggio macchina del 6502 restando bloccato di fronte alle sottigliezze dell'indiretto indicizzato e dell'indicizzato indiretto, trasecolerà ora alla notizia dell'apparizione di veri "mostri" quale l'indirizzamento stack relativo indiretto indicizzato o il diretto indicizzato indiretto lungo.

Per riassumere, anche a beneficio dei profani di linguaggio macchina, i nuovi modi di indirizzamento sono pensati per rendere molto più semplice, proficua e compatta la creazione di compilatori ed interpreti dei più nuovi e sofisticati linguaggi di programmazione.

10.6 Scrivere un programma per la famiglia Apple II

Per motivi commerciali potrà essere conveniente, sinché il parco macchine non si sia completamente rinnovato, scrivere programmi che possano operare sull'intera famiglia dei computer Apple II, e non solo sul più recente Apple IIgs.

La soluzione più semplice è quella di scrivere un programma per i vecchi modelli, controllare che funzioni correttamente su

Phoenix in modo emulazione e vivere felici. D'altra parte è un vero peccato non sfruttare in qualche maniera le capacità superiori della macchina.

A mio parere ci sono due modi di salvare capra e cavoli: il primo è commercialmente il più significativo. Nei punti in cui ha senso farlo, si identifica la macchina su cui si sta agendo (con la routine IDROUTINE descritta nell'appendice A) e ci si comporta in due modi diversi a seconda del risultato. Si tratta dell'espediente utilizzato dall'ultima versione di Apple Works, che se riconosce di trovarsi su Apple IIgs inoltra correttamente le chiamate al Memory Manager. Similmente, potreste trovare conveniente in taluni punti la scelta di aumentare la velocità, o di dirottare i calcoli matematici su SANE.

Il secondo modo è teoricamente migliore, in special modo se modo a sedici bit. Se ce n'è la possibilità commerciale, la seconda possibilità non va dimenticata: Apple stessa l'ha praticata anni fa, quando con l'uscita di Apple IIe mise in commercio due versioni differenti di Apple Writer II, la prima per il vecchio II+ e la seconda per il modello più recente.

10.7 L'interfaccia utente

Dal primo Apple IIc in poi il Mouse, introdotto per la prima volta nel mondo dei personal computer su Apple Lisa (la mamma di Macintosh), faceva la sua apparizione ufficiale anche nel mondo degli Apple II. Sono apparsi ben presto programmi che fanno uso dei *parafernalia* che si accompagnano usualmente al mouse: le finestre, i menù a discesa, gli sfogliatori e tutti gli altri oggetti software che compongono la cosiddetta **interfaccia iconica,** un modo consistente e coerente con cui i programmi possono dialogare con l'utente del computer.

L'interfaccia iconica ha una storia più lunga di quel che si pensa normalmente, ed è stata creata da un gruppo di esperti... diciamo pure scienziati... non solo di *computer science* ma anche di discipline dell'apprendimento, psicologia, ingegneria. Si tratta dunque di uno sviluppo ragguardevole nel campo del perso-

nal computing, e non va sottovalutata ma neppure sopravvalutata.

L'interfaccia iconica non ha sfondato completamente nel mondo degli Apple II per un semplice motivo: è piuttosto difficile da programmare. D'altra parte, con l'avvento dell'Apple IIgs, che incorpora routine belle e pronte per fornire l'interfaccia iconica, ogni resistenza sarà presto vinta: ogni programmatore bada a realizzare il suo prodotto, oltre che bene, anche in fretta quando possibile, e l'uso del toolbox è troppo semplice per essere scartato.

Due parole, allora, per programmatori ed utenti (ricordando agli interessati che il sottoscritto ha analizzato nei dettagli pregi e difetti dell'interfaccia iconica nel libro "conoscere i sistemi operativi", scritto in collaborazione con Carlo Lucio Bocchetti, anch'esso edito da Jackson).

L'interfaccia iconica è oggi di gran lunga la migliore che si possa implementare su un computer (nonostante qualche piccolo vizio): lo dimostra indirettamente la sua grande complessità che ha fatto sì che sia utilizzata solo dai più potenti e recenti personal computer, i cosiddetti "seconda generazione", mentre i modelli più vecchi (vedi PC IBM, guarda caso) hanno dimostrato di non poterne sopportare il costo computazionale.

Ciononostante, l'interfaccia iconica non è certo la migliore possibile, né la sua supremazia è destinata a restare per molto inviolata. Si fa già d'ora un gran parlare di sistemi esperti, di riconoscimento della voce e delle immagini e di altre migliorie.

Non posso dire quale forma prenderà l'interfaccia utente del futuro, ma sono disposto a fare una scommessa.

Apple II è stato il primo personal computer. E di gran lunga il più vecchio personal sul mercato, e tuttavia con questo ultimo modello il suo nome spicca anche nella lista dei quattro più potenti modelli (gli unici quattro, per l'appunto, di seconda generazione, nel momento in cui scrivo anche se sono certo che se ne aggiungeranno presto altri).

Quando verrà una nuova rivoluzione, Apple II sarà ancora e sempre tra i primi, perché le capacità di adattamento del gran vecchio sono di molti ordini di grandezza superiori a quelle di ogni rivale. Ecco la scommessa: tra vent'anni i computer saranno incredibilmente più potenti dei loro antenati di oggi, ed i nomi che circoleranno saranno totalmente diversi (pensate alle automobili anteguerra, per un pallido esempio). E tuttavia, sulla mia (e, spero, vostra) scrivania ci sarà un Apple II, che, dietro pre-

ghiera, farà apparire la scritta "APPLE DOS 3.3. NOW LOADING INTEGER BASIC INTO LANGUAGE CARD", e mi permetterà di giocare a Little Brick Out e a Star Trek, come nel 1977.

UNA INTRODUZIONE AL TOOLBOX

In questo capitolo espandiamo i concetti introdotti nel terzo paragrafo del capitolo 9.

11.1 Il toolbox

Abbiamo detto che il toolbox è un insieme di routine, di sottoprogrammi, che ogni programma può usare per svolgere compiti generali, come far suonare una nota o aprire una window (finestra) o gestire il mouse.
Per l'esattezza, nel toolbox sono presenti diverse centinaia di simili routine, raggruppate a seconda del loro uso: ad esempio tutte le routine che si occupano di window (apri una window, muovi una window, ingrandisci una window, chiudi una window eccetera) sono unite sotto il nome collettivo di Window Manager (manager delle finestre). I manager sono anche chiamati tool, cioè attrezzi, e presi tutti assieme formano il toolbox (scatola degli attrezzi).

La pur capace memoria ROM (permanente) dell'Apple IIgs non è in grado di contenere tutto il toolbox: una parte di esa viene allora caricata in memoria RAM quando accendete il computer con un dischetto ProDOS 16 inserito.
In memoria RAM vanno i tool di uso meno frequente, di modo che la memoria sia occupata solo quando ce n'è effettiva-

mente bisogno. Un esempio è il Note Synteshizer, un altro esempio è il Printer Manager che controlla la stampante.
In memoria RAM ci sono anche alcuni tool che sono stati sviluppati tardi dai programmatori della Apple Computers, che sono stati resi disponibili quando i computer Apple IIgs erano già in produzione, e che quindi non hanno trovato posto nello spazio permanente ROM. Un esempio è il Window Manager.
Infine, in memoria RAM vengono poste delle copie aggiornate e migliorate dei tool in ROM, se tali copie sono disponibili e se il programma che usa il toolbox lo richiede esplicitamente.
In un futuro molto vicino, molti altri tool verranno messi in memoria ROM (presumibilmente portandola al doppio della capacità) per poco prezzo: avverrà non appena la situazione dei tool si sarà stabilizzata, cioè quando saremo ragionevolmente sicuri che le versioni aggiornate sono state completamente liberate da errori di programmazione.

Un programma (anzi, una applicazione, come vorrebbe la terminologia ufficiale Apple) non ha modo di sapere quali tool, tra quelli che usa, sono in memoria RAM e quali in memoria ROM.

11.2 Uso del toolbox da Basic, Pascal e linguaggio C

Suppongo che molti dei miei lettori abbiano provato a programmare un poco, anche se probabilmente non conoscono l'Assembler. In questo breve paragrafo citiamo brevemente come avviene l'uso del toolbox per un programmatore Basic, Pascal, C o di altro linguaggio superiore.
Quando programmerete Apple IIgs avrete al fianco, insieme al manuale del linguaggio che usate per trovare i comandi del linguaggio, il manuale del toolbox con indicate tutte le routine dei tool. Quando avete bisogno di usare il toolbox per fare qualcosa, voi dovete solo supporre che esista una istruzione del vostro linguaggio che ha lo stesso nome della routine indicata nel manuale del toolbox.

Il toolbox viene richiamato con un metodo relativamente complesso dal linguaggio assembler (lo abbiamo accennato nel capitolo 9), ma un linguaggio di altro livello fa tutto il lavoro du-

ro per voi e vi permette di usare il toolbox come usereste delle normali istruzioni del linguaggio.

Facciamo un esempio: voi volete scrivere le parole "Ciao, mondo" sullo schermo grafico di altissima risoluzione.
Guardando sul manuale del toolbox osservate che esiste una routine chiamata "DrawString" che serve proprio a quello scopo. Allora scriverete in Basic...

```
100 DRAWSTRING "Ciao, mondo"
```

Mentre in Pascal scrivereste

```
DrawString ("Ciao, mondo");
```

Ed in linguaggio C la cosa diventerebbe

```
DrawString ("Ciao mondo\n");
```

Usando la routine del toolbox proprio come avreste usato il comando normale del linguaggio per scrivere una stringa (cioè rispettivamente "print", "writeln" e "printf").
In Basic alcune routine risulterebbero un po' più difficili da usare, perché usano delle variabili di tipo "puntatore": il tipo puntatore esiste e viene usato sia in Pascal che in C, ma nel Basic normalmente non esiste (per esempio, non esiste nel Basic Applesoft che è il più conosciuto ed usato nel mondo Apple).
Dato che non ho ancora visto nessun linguaggio Basic per Apple IIgs (gira voce che Microsoft stia preparando una versione del suo Basic, lo stesso che si usa su Macintosh) non so come la cosa verrà risolta: suppongo che la cosa più semplice sia introdurre il tipo puntatore anche in Basic.

Notate che non esiste alcun modo per usare le chiamate del toolbox con il Basic Applesoft: dovranno venire creati nuovi linguaggi per usare le capacità del toolbox.

11.3 Uno sguardo dentro il toolbox

Tutti i tool sono accomunati da alcuni principi generali: è necessario che tutte le routine lavorino in modo simile, per evitare ai programmatori di impazzire dovendo memorizzare le idiosincrasie di ogni singolo tool.
In particolare, tutte le chiamate di tutti i tool vengono invocati

con lo stesso metodo (quello della chiamata al Dispatcher, come abbiamo visto nel capitolo 9).

Prima di usare un qualsiasi tool bisogna chiamare il Tool Locator, un tool che è sempre presente in memoria ed è sempre attivo, di modo che questo possa caricare in memoria RAM i tool che non sono presenti.

Il Tool Locator viene invocato nel modo che dettaglio nelle prossime linee con una sintassi pseudo Pascal, che dovrebbe essere comprensibile a tutti:

```
Load tools: (Quanti tools: Integer;
             N di rif tool 1: Integer;
             Versione richiesta: Integer;
             N di rif tool 2: Integer;
             Versione richiesta: Integer;
             N di rif tool 3: Integer;
             Versione richiesta: Integer;
             ...eccetera...)
```

Questo significa che, se per esempio io sto scrivendo un programma che usa il Window Manager e il Menu Manager, e nessun altro tool, innanzitutto consulto la tabella che appare nel terzo paragrafo del capitolo 9, scoprendo così che il numero di riferimento del primo è 14 mentre il numero di riferimento del secondo è 15.

Poi scelgo quale sia la minima versione dei tool necessaria per il mio programma. Io so che il Window Manager conteneva un errore nella gestione dell'ingrandimento delle finestre, errore che è stato corretto nella versione 1.2. Allora imporrò che la minima versione per il Window Manager sia la 1.2. mentre per quanto riguarda il Menu Manager posso accontentarmi della versione 1.0 (la prima).

Scriverò allora

```
LoadTools (2, 14, $0102, 15, $0100);
```

ed il Tool Locator farà tutto quanto è necessario per mettermi a disposizione quei tool (io non ho bisogno di sapere che cosa venga fatto esattamente. Può darsi che vengano caricati dal disco, oppure può darsi che siano stati messi nella memoria permanente ROM e quindi siano già a disposizione: è irrilevante).

Se il Tool Locator non riesce a soddisfare la mia richiesta (per esempio perché ha a disposizione solo la versione 1.1 del Window Manager), me lo comunica con un codice di errore. Il mio programma può allora scegliere che azione compiere; normal-

mente un programma che non riesce a caricare i tool che gli servono si suicida: appare la mela che rimbalza sullo schermo con una scritta di spiegazioni per l'utente.

C'è un altro elemento che accomuna tutti i tool – ricordate: un tool è composto da molte routine, cioè da molti sottoprogrammi che compiono azioni elementari – comprendono delle routine chiamate:

BootInit (cioè "lancio")
StartUp ("inizio")
ShutDown ("chiusura")
Version ("versione")
Reset ("ripartenza")
Status ("situazione")

Per esempio, nel Window Manager, il tool che si occupa delle window e delle azioni che si possono compiere su di esse, esistono le routine WMBootInit, WMStartUp, WMShutDown, WMVersion, WMReset e WMStatus. Le prime due lettere indicano che quella routine particolare si applica al Window Manager: le analoghe routine del tool LineEdit hanno il nome che inizia con LE.

La routine BootInit viene eseguita dal Tool Locator quando gli viene comunicato che il programma attualmente funzionante ha bisogno di quel tool. Un programma non dovrà mai eseguirla direttamente.

La routine StartUp, invece, va eseguita dal programma. Serve a dare definitivamente "la sveglia" al tool richiesto.
È necessario che esistano due differenti routine, una di BootInit ed una di StartUp, anche se a prima vista parrebbe che le due possono essere unificate.
Infatti, alcuni tool hanno bisogno di uno spazio in memoria per salvarci i propri dati, in modo invisibile al mio programma e alle altre applicazioni che li usano. La routine di StartUp, al contrario di quella di BootInit, ha un parametro: il numero di identità della applicazione che usa il tool. Quando il mio programma viene portato in memoria per essere eseguito, gli viene assegnato un numero di identità differente dai numeri assegnati ad ogni altra applicazione presente in memoria, un numero convenzionalmente chiamato UserID.
Il mio programma memorizzerà in qualche modo quel numero e lo dovrà usare per comunicare con i tool: per riprendere

l'esempio fatto poco fa del programma che usa il Window e il Menu Manager, il programma eseguirebbe le istruzioni.

WMStartUp (UserID);
MMStartUp (UserID).

In questo modo i tool che hanno bisogno di memoria richiederanno la memoria che gli serve a nome del mio programma. Quando il mio programma terminerà, e la memoria che occupava sarà liberata per altri usi, verrà automaticamente liberata anche la memoria che i tool che il programma usava avevano richiesto.

Per alcuni tool la routine di StartUp potrebbe non avere nessun effetto, ma è comunque obbligatorio effettuarla, perché forse una versione futura di quello stesso tool ne avrà bisogno. Notate che tutti i tool restituiranno un errore se viene fatta loro una richiesta senza che sia stata impartita prima di tutto l'istruzione di eseguire uno StartUp.

La routine di ShutDown è complementare alla routine di StartUp. Mentre i programmi devono, come prima cosa, eseguire la richiesta di LoadTools e poi eseguire le chiamate di StartUp, dovranno anche come ultima cosa eseguire le chiamate di ShutDown. I tool eseguiranno, se del caso, delle operazioni che concludono il loro uso.

Pensate al caso del tool che permette di usare DOC: usando la routine di ShutDow il tool disattiverà il DOC, in modo da risparmiare energia e disinserire il lievissimo fruscio che DOC produce attraverso l'altoparlantino interno.

La routine Version serve ad identificare il numero di versione del tool. Abbiamo già spiegato cosa sia il numero di versione nel capitolo 7.

Grazie a questa routine il programma può rendersi conto di cosa ha a disposizione. Poniamo che io stia sviluppando un programma e mi renda conto che il tool Dialog Manager che sto usando contiene un errore in una delle sue routine: posso continuare a scrivere il mio programma e, usando la chiamata DMVersion, fare in modo che il programma usi quella chiamata solo se la versione è più recente di quella in mio possesso.

Notate che posso assicurarmi che il tool che sto usando non sia antecedente alla versione "X" di mia scelta, grazie alla funzione LoadTools che abbiamo visto poco fa: ma nel caso in esempio non posso imporre che venga usata una versione più recente di

quella che ho ora a disposizione: se lo facessi il mio programma si rifiuterebbe di venire usato e non potrei mai procedere nella sua codifica.

La routine di Reset viene usata per reinizializzare un tool. Serve solo in un numero limitato di casi, ma tutti i tool hanno una routine chiamata così anche se per molti la routine non fa nulla.
Non confondete la routine di reset con il tasto di reset presente sulla tastiera (il tasto con la serigrafia del trinagolo), che serve ad impartire un comando di ripartenza al microprocessore 65816.
La routine di reset serve, ad esempio, nel tool che controlla le operazioni del mouse per riportare il mouse nella posizione iniziale (nell'angolo in alto a sinistra dello schermo).
La routine di Status restituisce un valore booleano, cioè "falso" oppure "vero"; serve a scoprire se un dato tool è già stato attivato (con la chiamata di StartUp) oppure se è attualmente inattivo.

11.4 Gli eventi

Per comprendere esattamente come funziona un programma dentro Apple IIgs dobbiamo spendere due parole sul concetto di evento.
Per **evento** intendiamo qualunque azione svolgentesi nel mondo esterno che Apple IIgs è in grado di percepire e a cui può reagire nel modo più adatto.
La pressione di un tasto sulla tastiera è un evento, e la pressione di quello stesso tasto accompagnato da Mela Vuota è un evento leggermente differente.
Quando il mouse viene mosso, oppure il suo pulsante premuto, stiamo assistendo ad un evento.

L'accadere di un evento normalmente (il che vuol dire "sempre, tranne quando il programmatore esplicitamente sceglie di fare altrimenti") blocca l'esecuzione del programma per un breve e inavvertibile istante: il processore, sotto il controllo dello Event Manager, esegue una serie di azioni che trattano l'evento.
Questo è necessario perché, ad esempio, se il mouse non viene letto subito dopo essere stato mosso, le informazioni sul suo movimento vanno perse.

Lo Event Manager prende nota dell'evento e poi ritorna il controllo al programma che era stato interrotto.

Lo Event Manager tiene nota degli eventi che sono accaduti in una struttura chiamata "coda degli eventi": potete immaginarla come un casello dove gli eventi arrivati aspettano come automobilisti all'ingresso dell'autostrada.
Quando un programma decide che è venuto il momento di reagire agli stimoli esterni chiede all'Event Manager di comunicargli quale sia il primo evento in coda, lo identifica e lo tratta nella maniera che preferisce (la pressione del bottone di "OK" eccetera).
Un programma può anche decidere di trattare solo un certo tipo di eventi: per esempio se un programma sceglie di non usare il mouse può chiedere all'Event Manager di comunicargli quale sia il primo evento che non coinvolga il mouse. In quel caso gli eventi creati dal mouse resteranno in coda e saranno sorpassati dagli altri.
Ovviamente, se la coda degli eventi è vuota (perché l'utente del computer è andato a bersi un caffé), lo Event Manager restituisce un valore speciale che significa "nessun evento è accaduto".

11.5 La programmazione basata su eventi

L'esistenza dell'Event Manager e il modo particolare in cui questo tratta le risposte dell'utente (lo input) ha delle coseguenze sul modo di programmare che viene usato su Apple IIgs e sugli altri personal computer simili, come Macintosh.

Quando voi scrivete un programma in modo tradizionale voi imponete in alcuni punti all'utente di battere delle risposte alla tastiera: in un programma tradizionale è comune vedere delle istruzioni del tipo...

```
1100 INPUT "Qual è il tuo nome?"; NO$: REM Basic
```

```
Write ('Qual è il tuo nome?');
ReadLn (nome); (*Pascal*)
```

Quello che accade in un programma ben fatto con interfaccia

iconica è che in nessun punto il programma si interrompe per aspettare una risposta dell'utente.

Quando al programma interessa avere input dell'utente si limita a chiamare Event Manager per controllare se un tale input è già stato dato.

Il mouse si muove in uno spazio pieno di possibilità: in alto c'è la sbarra dei menu che permette di selezionare molte opzioni, più in basso molte finestre che possono essere allargate e rimpicciolite, chiuse e riaperte. Ci possono essere dei controlli (i bottoni su cui fare click col mouse, sbarre di scorrimento ed altro ancora) e punti in cui è possibile scrivere.

Il programma si limita a fornire dei sottoprogrammi (le procedure) che trattano ciascuna di queste possibili azioni dell'utente: non esiste propriamente una parte principale del programma che chiama quelle procedure quando pare che sia il caso.

Per l'esattezza, diremo che una parte principale del programma esiste sempre: specifica che cosa fa il computer in attesa di un evento (per esempio può suonare un motivetto), in quali momenti vanno trattati gli eventi, e specifica anche il momento in cui il computer è libero di trattare gli eventi esterni al programma. Quest'ultima affermazione si riferisce agli accessori di scrivania: quando un evento si riferisce ad un accessorio, il programma principale, (la applicazione), non ne è cosciente, non riceve alcun avviso dallo Event Manager.

Quando la applicazione decide che è tempo di lasciare spazio agli accessori di scrivania, questi prendono il controllo e ricevono dallo Event Menager le informazioni relative all'evento che li riguarda.

12

IL TOOLBOX

In questo capitolo dedichiamo un paragrafo a ciascuno dei tool più importanti.

In questo modo scopriremo quali sono le possibilità aperte al nostro computer, quali i suoi punti di forza e quali le (poche) debolezze.

Per evitare confusioni, prima di iniziare, vorrei dare le definizioni di alcune entità: con *utente* intendiamo la persona che userà il computer con i programmi sviluppati da qualcuno, e che non ha necessariamente bisogno di sapere come i progrmmi lavorano. Con *programmatore* intendiamo una persona che crea le applicazioni (i programmi) che l'utente userà.

Talvolta parlo delle applicazioni o dei tool in modo personale: "il QuickDraw allora reagisce così e così" è semplicemente un modo più leggibile per dire che "il QuickDraw è programmato in modo da reagire così e così".

12.1 QuickDraw II

QuickDraw II è il più grande e il più importante tool, sia per dimensioni che per numero di funzioni messe a disposizione. QuickDraw può eseguire in tutto circa duecento diverse operazioni, tutte nall'ambito del display grafico.

QuickDraw II è basato sul suo omonimo e corrispettivo, il

QuickDraw che troviamo nel toolbox di Apple Macintosh. Se non consideriamo alcune differenze marginali e la velocità (il QuickDraw di Mac è considerevolmente più veloce di QuickDraw II) l'unica differenza sta nel fatto che Apple IIgs permette il display a colori, mentre il Macintosh è un computer in bianco e nero: dunque sono state aggiunte delle chiamate per trattare convenientemente il colore. Sarà interessante scoprire se, con l'avvento di Macintosh II (si, il nome è un omaggio all'Apple II), che introduce il colore anche nel mondo Mac, QuickDraw/Mac verrà modificato in modo da somigliare ancora di più a quello di Apple IIgs.

QuickDraw II contiene più di 200 differenti routine. La maggior parte di QuickDraw II si trova in memoria permanente, la ROM, ma alcune routine del livello più alto (cioè le più sofisticate e quelle usate più di rado) sono state raggruppate a parte e vengono caricate in RAM quando ce n'è bisogno. Questa espansione a QuickDraw II prende il nome di Auxiliary QuickDraw.

QuickDraw II disegna sullo schermo di super alta risoluzione dell'Apple IIgs, che come abbiamo visto ha una risoluzione di 640 x 200 pixel (punti): tuttavia, in se QuickDraw II non è limitato a quelle dimensioni: lo schermo virtuale sul quale esso può agire ha le dimensioni di 32767 x 32767 pixel. In questo modo è garantita la conpatibilità con i futuri modelli della serie Apple II che, presumibilmente, introdurranno risoluzioni video anche più elevate.

Per la cronaca, i ricercatori giapponesi nel campo della computer graphics hanno come obbiettivo la risoluzione di 10000 x 10000 pixel entro il 1992. La attuale risoluzione di Apple IIgs, i 640 x 200 pixel, fornisce una nitidezza dell'immagine pressappoco analoga alla risoluzione della televisione. Se fosse possibile arrivare a vedere 1000 x 1000 pixel sullo schermo si sarebbero superati i limiti della fotografia e della cineripresa tradizionale (a 35 millimetri).

QuickDraw II fornisce i mezzi per trattare con oggetti, con entità astratte piuttosto sofisticate: le icone (i disegnini che rappresentano oggetti sullo schermo grafico e che possono venire spostati con il mouse), linee, rettangoli, ovali, archi di cerchio, poligoni (figure delimitate da segmenti), rettangoli con gli angoli arrotondati, testo (caratteri alfabetici, numeri e simboli rappresentati in più stili tipografici), e "immagini", termine con il quale intendiamo una regione dello schermo che rappresenta qualche

cosa. Una fotografia digitalizzata, per esempio, è una "immagine".

Le routine incorporate in QuickDraw II sono raggruppabili per funzione: ci sono le routine di controllo, come in tutti i tool, cioè le routine di BootInit, StartUp, Status eccetera che abbiamo presentato nel precedente capitolo. Vengono poi le routine cosiddette "globali" perché agiscono sui parametri di tutto ciò che appare sullo schermo: in questo gruppo troviamo le chiamate che predispongono i colori da usare.

In un terzo gruppo, le chiamate GrafPort, viene trattato il posizionamento dello schermo. Come abbiamo detto lo schermo logico sul quale Phoenix, via QuickDraw II, disegna è molto più ampio dello schermo video. Lo schermo video con i suoi 128.000 punti viene pensato come una finestra su quello schermo, lo schermo logico, che è molto più grande.

Per comprendere questo concetto, prendete una grande fotografia: rappresenta lo schermo logico, lo spazio di 32000 × 32000 punti nel quale QuickDraw disegna.

Poi prendete un foglio di carta bianca, e ritagliatevi uno spazio rettangolare all'interno, buttando via il rettangolino ritagliato e tenendo il foglio con il buco. Se poggiate il foglio bucato sulla fotografia, potete guardare la grande fotografia attraverso il buco: potete guardarne una parte qualunque facendo scivolare il foglio bianco sulla superficie. Allo stesso modo, QuickDraw usa lo schermo video per visualizzare una parte del suo grande foglio da disegno: lo schermo del vostro monitor è il foro del foglio di carta che vi permette di vedere una piccola parte della grande fotografia.

Un gruppo di routine tratta la penna di QuickDraw. Quando QuickDraw è chiamato a tracciare una linea può farlo con uno spessore scelto a piacere e in un colore scelto a piacere: con le routine di questo gruppo la penna viene dimensionata come è più utile.

La penna immaginaria di QuickDraw può disegnare in bianco, in nero, in un colore a scelta, o anche in più colori insieme, secondo un *pattern*. E dunque possibile avere una magica penna che disegna lasciando nella sua scia righe alternate in bianco e nero o altre tessiture più complicate.

Un quinto gruppo di chiamate serve a trattare con le fonti di caratteri: questo gruppo di chiamate interagisce con un altro tool del toolbox, il Font Manager, per creare gli stili (neretto,

sottolineato, corsivo, ombreggiato eccetera) e per ingrandire o rimpicciolire i caratteri.

Il sesto gruppo, il più nutrito, comprende tutte quelle routine che permettono di disegnare. Gli oggetti disegnabili con una sola chiamata vanno dalle più semplici linee ai più complessi poligoni riempiti di un colore o di un pattern a scelta. In questo gruppo si trova un sottogruppo di chiamate per controllare il cursore, che può venire nascosto, reso visibile ed invisibile, e cambiato ad una qualunque forma.

12.2 Il Memory Manager

Memory Manager è il padrone assoluto della memoria RAM contenuta nel computer.

Quando il sistema operativo ProDOS deve caricare in memoria una applicazione, è al Memory Manager che chiede la memoria necessaria. Quando un programma ha bisogno di uno spazio per i suoi dati, è Memory Manager che lo concede. E quando è necessario effettuare una stampa, è il Memory Manager a fornire i mezzi perché la stampa non blocchi il computer, permettendo che una zona di memoria venga assegnata al chip schiavo di stampa senza obbligare il 65816 a passargli i dati un byte la volta.

Memory Manager distingue tra la memoria normale e quella privilegiata, cioè il banco zero di memoria (indirizzi tra 00/0000 e 00/FFFF) che è indispensabile per i più complessi modi di indirizzamento del microprocessore 65816.

Sino da oggi, Memory Manager sa che un Apple IIgs può contenere sino a 16 megabyte di memoria ed è preparato a gestirla. (Talune zone sono riservate: abbiamo già visto la mappa di memoria di Apple IIgs con le distinzioni interne).

Dato che ovunque sia necessario memorizzare un indirizzo di memoria il Memory Manager riserva quattro byte di spazio, anziché i soli 3 byte che sarebbero necessari per un numero compreso tra zero e 16 mega, in futuro sarà possibile raggiungere il quantitativo di 4 gigabyte, cioè circa 4 miliardi e 300 milioni di byte.

Quando Memory Manager concede uno spazio in memoria ad un richiedente, memorizza che quello spazio è occupato – in

modo che non sia possibile riassegnarlo ad un altro programma creando conflitti di interesse in memoria – e segna anche il numero d'identità dell'applicazione richiedente, di modo che quando quella avrà terminato il proprio funzionamento sarà possibile liberare tutta e sola la memoria che occupava.

Quando una applicazione termina, il Memory Manager mette i blocchi di memoria che ne contenevano l'immagine eseguibile in uno stato chiamato zombie. Se la memoria libera si esaurisce, il Memory Manager passerà ad assegnare anche i blocchi zombie: ma se prima di quel momento accadesse che l'utente sceglie di riprendere ad usare quella stessa applicazione il Memory Manager in accoppiata con il Relocating Loader si renderebbe conto che una copia di quel programma è già in memoria, e lo eseguirebbe immediatamente senza ricaricarlo da disco, senza che il ProDOS 16 o l'utente debbano rendersi conto della differenza di trattamento. È una trovata molto brillante, che spesso si rende comoda: forse avevate già notato che spessissimo, quando una applicazione termina, il Program Launcher viene riesumato istantaneamente senza ricaricarlo dal disco.

I pezzi di memoria sono assegnati ai programmi richiedenti che ne fissano anche il valore. Per esempio, una immagine caricata dal disco per essere visualizzata non è molto preziosa, perché in caso di necessità può venire ricaricata dal disco; al contrario, dei dati appena introdotti dall'utente sono molto preziosi e non vanno assolutamente danneggiati.

Se per caso la memoria libera comincia a mancare, il Memory Manager si libera dapprima dei pezzi di memoria zombie, e poi dei dati non indispensabili in ordine via via crescente di valore, ma non distrugge mai i dati che sono stati indicati come vitali: nel caso dell'esempio potrebbe essere forzato a liberarsi dell'immagine, ma mai e poi mai potrebbe cancellare i dati introdotti. I lettori più ferrati in informatica riconosceranno in questo schema i principi della virtualizzazione della memoria.

Ci sarebbero altre cose da dire sul memory manager: in particolare, i metodi che esso usa per la ricompattazione della memoria. Si tratta comunque di metodi classici, che sono descritti in qualunque buon libro sui sistemi operativi, e dunque non li descriveremo qui.

12.3 I Control e Dialog Manager

Il Control Manager è il tool che permette l'uso di tutti gli attrezzi manovrabili con il mouse, e cioè principalmente i bottoni e le sbarre di scorrimento (chiamate anche, impropriamente, sfogliatori). Control Manager permette di disegnare anche dei controlli originali e fornisce i mezzi per usarli come quelli predefiniti.

Ad esempio, è piuttosto semplice definire un oggetto a forma di termometro che indica quanta memoria è rimasta libera entro il computer: il livello del "mercurio" salirà quando la memoria si avvia a terminare.

Il Dialog Manager è strettamente imparentato con il Control Manager (e con il Window Manager che vedremo tra poco): è quel tool che fornisce le finestre di errore, di avviso e di colloquio con l'utente.

Per finestre d'errore e di avviso intendiamo parlare di quegli spazi che appaiono indicando i pericoli (come "Attenzione: vuoi salvare il lavoro fatto sin qui prima di Annullare?") o le situazioni speciali che il programma non può risolvere senza l'aiuto dell'utente (come "Errore: il disco è protetto dalla scrittura. Sposta la protezione.").

Per usare questi mezzi standard di comunicazione è sufficiente indicare al Dialog Manager il messaggio che deve apparire, e il Dialog Manager farà apparire il messaggio con la icona più adatta e i bottoni che possono servire (qualche volta solo "OK" per confermare di aver letto il messaggio, come nell'esempio della protezione dalla scrittura, qualche volta una scelta, come nell'esempio dell'azzeramento di un programma).

Ci sono quattro icone già definite per questo tipo di comunicazione. L'icona "Errore" è l'omino con il fumetto che contiene un asterisco. L'icona "Attenzione" è un cartello triangolare con un punto esclamativo all'interno. L'icona "Stop" è un cartello ottagonale con una mano aperta nel centro. L'icona "Nota che..." è l'omino con il fumetto che parla.

Oltre ai mezzi per inviare messaggi all'utente, il Dialog Manager, come il suo nome indica, permette di ricevere ordini colloquiando con l'utente del computer. Un esempio classico è quello spazio pieno di bottoni e spazi per la risposta che appare scegliendo Page setup in un word processor, oppure la schermata

per scegliere un programma che vedete nel Program Launcher: sono tutte creazioni del Dialog Manager.

12.4 Il Resource Manager, che non c'è

Due parole, prima di continuare, su un tool che non esiste e non ci sarà mai in Phoenix, ma che c'è e condiziona pesantemente il modo di agire dentro il fratellone Apple Macintosh.

Per facilitare la traduzione dei programmi nelle varie lingue nazionali è stato introdotto nel Macintosh un Manager chiamato "Resource Manager", il controllore delle risorse. L'idea sarebbe che le scritte, le icone e tutto quanto viene fatto apparire sullo schermo – comprese le definizioni delle finestre di dialogo del Dialog Manager – in breve tutti i dati del programma sono conservati separatamente dal codice del programma, e per accedervi il programma fa delle richieste al Resource Manager.

I progettisti di Apple IIgs hanno deciso di non usare il Resource Manager. Questo significa che è più difficile creare una versione nazionale di un programma (non è impossibile come era sui vecchi Apple II, perché i dati sono pur sempre separati dal codice e memorizzati in un apposito segmento che il Relocating Loader e il Memory Menager riconoscono), ma d'altra parte è possibile realizzare una struttura più flessibile nel programma.

E molto difficile spiegare questo punto a chi non ha conoscenze approfondite di teoria informatica: se il mio lettore non ha le conoscenze per comprendere la spiegazione che segue, dovrà semplicemente accettare in fede la precedente affermazione sulla flessibilità di Phoenix.

L'idea è che con Macintosh, data la presenza del Resource Manager, le risorse debbono essere create staticamente, prima della compilazione. Su Apple IIgs, invece, è possibile creare risorse sul momento, quando ce n'è bisogno, in modo dinamico, oltre ad usare le risorse statiche.

Data l'assenza del Resource Manager, il Dialog Manager e il Control Manager di Phoenix sono sostanzialmente differenti dai loro corrispettivi su Macintosh.

12.5 Il Window Manager

Il Window Manager, più volte citato come esempio in questo libro, è uno tra i tool più importanti ed anche, a modesto parere di chi scrive, il meglio riuscito. Mentre per molti versi quasi tutti i tool di Apple IIgs sono stati creati per rispecchiare o imitare quelli preesistenti di Apple Macintosh, il Window Manager non si pone problemi nel sopravanzare di molti ordini di grandezza il modello.

Window Manager permette la creazione e al gestione delle cosiddette finestre, gli spazi sul video che possono venire ingranditi e rimpiccioliti, spostati, chiusi e riaperti eccetera.
Le finestre sono un trucco molto ingegnoso per consentire all'utente di manipolare più informazioni di quanto lo schermo possa visualizzare in una volta sola; e sono un mezzo concettualmente nitidissimo per separare in aree differenti del display i differenti compiti.

Alla gestione delle finestre, Window Manager affianca una funzionalità in più: la singola routine più potente ed utile dell'intero toolbox, lo stupefacente TaskMaster, cui dedicheremo il prossimo paragrafo.

Una finestra per Phoenix è uno spazio di qualunque forma sullo schermo (nulla vieta di usare una finestra circolare o a forma di mela). A seconda delle necessità, a scelta del programmatore, la finestra può godere di una serie di controlli:

* Una cornice con il nome della finestra
* Un bordino (usato nelle finestre del Dialog Manager)
* Un bottone di chiusura in alto a sinistra
* Un bottone di crescita in alto a destra
* Una sbarra di scorrimento orizzontale, in basso
* Una sbarra di scorrimento verticale, a destra
* Un punto per l'allargamento in basso a destra
* Una sbarra informativa sotto il titolo

Quest'ultima caratteristica merita due parole: introdotta proprio in Phoenix, la sbarra informativa può avere una dimensione scelta dal programmatore, e può anche contenere una sbarra di menu a discesa alternativa a quella principale che si trova in alto nello schermo. In questo modo un accessorio di scrivania può

disporre della sua sbarra di menu senza dover disturbare quella principale dell'applicazione.

12.6 TaskMaster

Una delle più grandi scocciature per un programmatore è, con la interfaccia iconica come in ogni altro ambiente, la gestione dei dettagli marginali: dover scrivere del codice che tratta le operazioni intuitivamente elementari distrae il programmatore dal suo compito principale e dalla codifica dei compiti veramente significativi. Su Phoenix, grazie a TaskMaster, gran parte del peso dovuto alle banalità viene tolto dalle spalle del programmatore.

Vediamo cosa è necessario codificare in una applicazione che usa alcune finestre in un normale computer dotato di Toolbox; userò una forma a pseudo programma per farmi seguire anche da chi non conosca nessun linguaggio di programmazione.

```
Attiva Le Finestre;
Attiva I Menu;
Attendi Una Azione;
Che Azione E?
  Se Ha Premuto Il Bottone Del Mouse
    Se E In Una Finestra
      WindowManager: Dov'E?
      Se E Sul Bottone Di Chiusura
        WindowManager: Chiudi La Finestra
      Se E Sul Bottone Di Zoom
        Window Manager: Allarga La Finestra
        Se E Sulla Sbarra Di Scorrimento
          Window Manager: Fai Scorrere
        Se E Sulla Sbarra Dei Menu
        Menu Manager: Lascia Che Scelga
        Se Ha Scelto Un Mio Menu
          Esegui l'Azione
        Se Ha Scelto Un Accessorio
          Desk Manager: Esegui l'Accessorio
        (eccetera)
      Se E Su Una Icona
      (eccetera)
```

Se Ha Mosso Il Mouse
Se Stava Spostando Una Icona
QuickDraw: Cancella La Vecchia Icona
QuickDraw: Ridisegnala Nel Nuovo Punto
Se Stava Spostando Una Finestra
(eccetera)
Altrimenti
QuickDraw: Muovi Il Cursore
Se Ha Premuto Un Tasto
Se E Un Equivalente Di Un Menu
Menu Manager: Tratta Il Menu
(eccetera eccetera).

Avrete notato che lo scopo di tutto questo è piuttosto evidente: far funzionare i tool in modo coerente ed unitario.

Ecco dove arriva TaskMaster: invece di dover scrivere tutto il codice riportato sopra; un programmatore piò limitarsi a chiedere:

Attiva Le Finestre;
Attiva I Menu;
TaskMaster (PensaciTu);

TaskMaster svolge tutti i compiti usuali (ma è anche possibile fare in modo che qualcuno di questi compiti venga svolto dalla applicazione se ce n'è bisogno, per esempio per gestire in modo inusuale un oggetto sullo schermo).

Task Master svolge tutto il lavoro noioso, e quando si trova di fronte ad una richiesta significativa la passa alla applicazione. Ad esempio, comunica le voci scelte nei menu all'applicazione in modo che questa possa svolgere per ciascuno di questi la azione appropriata.

E non bisogna pensare che TaskMaster si limiti ai compiti semplici: ad esempio, è in grado di trattare correttamente (con l'aiuto dello Scrap Manager, il tool che occupa dell'Archivio Appunti) il taglia-e-incolla anche tra differenti applicazioni, oppure tra una applicazione e un accessorio di scrivania.

Task Master, oltretutto, svolge i suoi compiti più velocemente di quanto potrebbe fare una applicazione, perché essendo una routine del toolbox può permettersi di prendere alcune scorciatoie nell'invocare le altre routine.

L'uso di Task Master, infine, garantisce che se nel futuro vi saranno migliorie nei tool – con l'introduzione di nuove specifiche e nuove possibilità – sarà automatico per le vecchie applica-

zioni sfruttare quelle possibilità in più: sarà Task Master, aggiornato per usare quelle ulteriori capacità, a trattarle per conto della applicazione.

12.7 Il Menu Manager

Lo scopo di Menu Manager è piuttosto ovvio: fornire i mezzi per la creazione e la gestione dei menu a discesa, quelli usati normalmente e che troviamo in alto sullo schermo, e quelli speciali introdotti dal Window Manager che appaiono nelle finestre.

Una applicazione chiama il Menu Manager perché la sbarra dei menu venga creata, comunicandogli i titoli dei menu e le singole voci che vi appaiono. E possibile anche usare una fonte di caratteri non standard, un colore differente dal tradizionale bianco e nero, una altezza della sbarra differente ed altre caratteristiche meno importanti. Le singole voci dei menu possono apparire in nero (selezionabili) o in grigio (disinserite) e con o senza il segno di spuntatura al fianco. E possibile inserire i segni di divisione tra una voce e l'altra di un menu.

Il Menu Manager ha anche la possibilità di trattare autonomamente gli alias, ovvero i tasti comando – mela vuota più una lettera – che vanno considerati analoghi alla scelta di alcune voci dei menu. Quando Event Manager comunica che è stato premuta la combinazione, Menu Manager su richiesta dell'applicazione scorre la sua tabella di alias e comunica a quale voce di un menu quel comando equivale. Ovviamente, l'applicazione non deve scomodarsi ad eseguire direttamente questa serie di chiamate ad Event e Menu Manager se sta usando TaskMaster.

La cosa interessante da esaminare è la capacità del Menu Manager di trattare autonomamente due dei menu, stilando da sè la lista dei nomi che dovranno apparirvi.

In collaborazione con il DeskManager, Menu Manager gestisce il menu mela (il primo menu in alto a sinistra): è sufficiente invocare la routine di FixAppleMenu perché appaiano entro il menu contrassegnato dalla melina colorata i nomi di tutti gli accessori di scrivania disponibili.

Se l'utente sceglie un accessorio da quel menu, Menu Mana-

ger non lo comunica all'applicazione ma cede direttamente il controllo all'accessorio.

Analogamente viene trattato il menu fonti, in collaborazione tra Menu Manager ed il Font Manager; con una chiamata a FixFontMenu – che è una chiamata del tool Font Manager, non di Menu Manager come la analoga FixAppleMenu – è possibile avere in un menu scelto dal programmatore dell'applicazione l'elenco di tutte le fonti di caratteri disponibili al momento.

12.8 Il LineEdit

Un altro tool significativo è il LineEdit, il tool che gestisce l'introduzione di testo in uno spazio sul video grafico usando insieme il mouse e la tastiera.

Con mio sommo rammarico debbo confessare che LineEdit è un'altra vittima, come QuickDraw II, della filosofia autodistruttiva "facciamo tutto un po' peggio che su Macintosh". Mentre l'analogo tool di Macintosh permette di trattare come una unica unità logica molte linee di testo, LineEdit vede solo una linea di caratteri la volta, con l'unica eccezione di una chiamata, LETextBox, che comunque non riconosce la giustificazione frontale delle linee, cioè l'aggiustamento delle parole in modo che siano allineate sia a destra che a sinistra nel testo.

Non c'è dubbio che LineEdit sia un gigantesco balzo avanti rispetto a GetLn, la routine facente parte di System Monitor che veniva fornita sui vecchi Apple II per consentire l'introduzione di una linea di testo dalla tastiera. D'altra parte, LineEdit così com'è è insufficiente alla maggior parte delle applicazioni serie, come i word processor iconici – i cosiddetti wysiwyg, what you see is what you get, cioè "vedi sullo schermo ciò che apparirà sul foglio stampato".

L'unica consolazione è che LineEdit non è uno dei tool che si trovano nella memoria permanente ROM, e dunque sarà particolarmente semplice e conveniente sostituirlo in futuro con una versione più adeguata al livello di sofisticazione del resto del toolbox.

Dato che la maggior parte dei programmi che permettono l'elaborazione dei testi useranno LineEdit, e dato che tutte le finestre di comunicazione del Dialog Manager usano LineEdit, tanto

vale spendere qualche parola sulle caratteristiche dell'editore di linea.

Una sbarra verticale lampeggiante indica il punto di inserimento: se battete dei tasti i caratteri premuti appariranno in quel punto. Il cursore di inserimento è spostabile usando il mouse oppure usando i tasti freccia destra e sinistra – che si trovano in basso a destra sulla tastiera.

La pressione dei tasti freccia muove il cursore di un carattere. La pressione dei tasti freccia insieme al tasto Option sposta il cursore di una parola. Battere le frecce insieme al tasto mela vuota sposta il cursore all'inizio ed alla fine della linea.

Con il tasto Delete viene cancellato l'ultimo carattere battuto, quello che si trova all'immediata sinistra del cursore. Usando invece Control–F viene cancellato il carattere alla immediata destra del cursore. Come avviene anche in AppleWorks, Control–Y cancella tutto quanto si trova alla destra del cursore sino al termine della riga.

Il tasto Clear (il tasto con un rettangolo cancellato che si trova in alto a sinistra nel tastierino numerico) cancella tutta la linea che si stava battendo, comprendendo sia i caratteri a destra, sia quelli a sinistra del cursore.

E possibile selezionare un brano del testo; il brano selezionato è distinguibile dato che viene disegnato con i colori invertiti rispetto al resto. Quando un testo è selezionato può essere tagliato ed incollato, oppure cancellato con la pressione del tasto Delete.

Un pezzo del testo può venire selezionato posizionando il cursore all'inizio del pezzo, premendo il pulsante del mouse e spostando il mouse – tenendo il pulsante premuto – sino alla fine del pezzo che si vuole selezionare.

È possibile selezionare una intera parola posizionandosi su un qualunque carattere di questa e facendo un doppio click con il mouse; un triplo click seleziona tutta la linea.

Se non si vuole usare il mouse, i tasti freccia selezionano anziché spostare il cursore se sono accompagnati dal tasto shift (uno dei due tasti identici contrassegnati dalla freccia verso l'alto).

Shift-frecce seleziona un carattere; shift-option-frecce seleziona una parola; shift-mela vuota-frecce seleziona sino all'inizio o al termine della linea.

12.9 I tool del suono

Come abbiamo visto nel nono capitolo, esistono tre tool dedicati alla gestione delle stupefacenti capacità musicali di Apple IIgs. Essi costituiscono una piramide, basata sul più semplice Freeform Player, il tool numero 8 che si trova nelle ROM di Apple IIgs. Permette maggiori sofisticazioni, salendo di uno scalino nella piramide, il Note Syntesizer. Per applicazioni molto professionali ed insuperabili, sarà presto disponibile (probabilmente sarà già in circolazione quando entrerà in vendita questo libro) il Note Sequencer, un vero direttore d'orchestra.

Per comprendere come possano agire i tool del suono, oltre alla spolverata di informazioni hardware del primo capitolo, abbiamo bisogno di qualche conoscenza di base sulle onde e il suono.

Come probabilmente saprete, un suono è una onda che viaggia nell'aria: è fatto di compressioni e rarefazioni dell'aria che, viaggiando dal punto d'origine del suono, giungono al nostro orecchio dove stimolano una membrana ed alcuni ossicini che, a loro volta, creano le informazioni nervose che noi percepiamo.

Una nota pura è un'onda di forma sinusoidale. Suoni particolari sono ottenuti con le onde triangolari e con quelle quadrate – tanto per saperlo, il suono prodotto dai vecchi Apple II e quindi anche da Apple IIgs quando agisce in modo emulazione, è composto unicamente da onde quadre.

L'altezza dell'onda (l'asse y nel grafico della funzione seno) è il volume del suono. L'ampiezza di un periodo completo (una oscillazione completa sull'asse x) è chiamato "periodo" oppure "ciclo". L'asse x rappresenta il tempo: il periodo dell'onda è misurato in cicli per millisecondo, il numero di volte in cui l'onda fa un *su* e *giù* completo in un millesimo di secondo. L'unità di misura "ciclo per millisecondo" è chiamata Hertz, abbreviata in Hz.

Per esempio, il do di mezzo di un piano ha una frequenza di 264 Hz, ovvero l'onda vibra e completa il suo periodo 264.000 volte ogni secondo.

Phoenix tramite il suo DOC può generare onde di qualunque forma: anzi, ne può generare 30 contemporaneamente ed indipendentemente.

Il suono prodotto da un computer è basato su una rappresen-

tazione di un onda sonora conservata nella memoria del computer. Tutti noi sappiamo che il computer ha una memoria composta di bit, raggruppati in byte, una memoria rappresentabile come una serie di informazioni binarie : 0111001010011100... eccetera.

Quando memorizziamo un onda sonora in un computer, parliamo di rappresentazione digitale, perché è basata sui numeri (in inglese, digit, dal latino digitus). La rappresentazione digitale di un suono è potenzialmente la più perfezionata possibile: è la stessa usata nei compact disk. D'altra parte, occupa un sacco di spazio in memoria (si può risparmiare spazio se si desidera una rappresentazione povera di particolari): pensate che un compact disk, capace di circa cento minuti di suono, è un disco dalla capacità di 600 megabyte. Quasi mille volte più della capienza di uno dei dischetti da tre pollici e mezzo dell'Apple IIgs.

In un nastro magnetico come quello dei registratori, il suono è memorizzato in forma analogica. Molto grossolanamente potremmo dire che la onda sonora viene trasformata in una forma manipolabile, un impulso magnetico, che impressiona il nastro. Il nastro può poi venire riletto da una testina che viene sollecitata dall'impressione, e l'impulso così ottenuto viene riconvertito in onda sonora.

C'è sempre della perdita di qualità, in questo modo, dato che l'impulso ottenuto alla lettura è solo una approssimazione di quello originale che era stato registrato.

Ovviamente, è possibile trasformare il suono analogico in digitale e viceversa. Phoenix ha, già incorporati, i mezzi per effettuare un campionamento, come abbiamo accennato nel primo capitolo.

Quando si vuole studiare il suono prodotto da uno strumento musicale, si possono individuare facilmente quattro momenti distinti.

In un primo momento il suono si crea dal nulla, fino a raggiungere il massimo volume. Subito dopo, il volume del suono scende sino a raggiungere un valore medio che viene mantenuto per un tempo relativamente lungo. Infine, il suono scema sino a svanire.

I quattro momenti prendono i nomi (nell'ordine) di Attack, Decay, Sustain e Release, e vengono normalmente indicati insieme come ADSR, dalle iniziali.

Conoscere lo ADSR di uno strumento è indispensabile per imitarne e riprodurne il suono.

Eccoci allora ai tool.

Il Freeform Player (tool n 8) svolge i compiti più semplici e legati allo hardware. Permette di mettere una campionatura digitale, magari letta dal disco, dalla memoria principale alla speciale memoria riservata a DOC. Oppure, se DOC è stato usato per campionare un suono, è possibile effettuare il trasferimento inverso e trasportare i dati ottenuti nella memoria principale in modo che possano essere elaborati e salvati.

Una volta che DOC è stato foraggiato con una campionatura, il tool Freeform Player svolge un compito molto importante. Abbiamo detto che Apple IIgs tramite il DOC può generare 30 voci, una per ogni oscillatore di DOC – altri due oscillatori non sono usati per produrre suono ma per altri compiti di gestione dentro il computer. Il Freeform Player permette una prima sofisticazione: accoppia i 30 oscillatori in modo da ottenere 15 voci stereofoniche. Per la cronaca, una coppia di oscillatori è chiamata generatore di suono.

Il tool Note Synthesizer (tool numero 25) tratta con oggetti che sono chiamati **banche di dati delle onde**. Una volta che sia stato campionato il suono di uno strumento, facendo "sentire" a DOC il suono di tutte le note e costruendo una tabella costituita dai dati così ottenuti, è possibile dare in pasto la tabella al tool che permette di accoppiare una tabella ad ogni generatore, trasformando il computer in una orchestra di 15 elementi.

Sia detto per inciso, pare che Apple Computers metterà presto in vendita a poco prezzo agli sviluppatori di software una biblioteca di suoni raccolti in tabelle per l'uso con il Note Synthesizer. Alcune centinaia di strumenti tra cui scegliere: qualcosa che potrebbe destare l'invidia dei direttori d'orchestra in carne ed ossa.

Infine, arriva il tool Note Sequencer (tool n 26). Questo permette di controllare la miscelatura dei suoni, le pause, i semitoni e gli altri elementi che costituiscono una musica. Inoltre, Note Sequencer permette di sincronizzare il microprocessore 65816 con DOC, di modo che sia possibile continuare ad usare il computer mentre DOC sta suonando e far sentire i suoni esattamente quando vogliamo.

Personalmente, non vedo l'ora di... sentire qualche videogioco!

13

LE MANI SU PHOENIX

Prima di concludere, ancora poche note per i programmatori: qualche notizia per quanti hanno già programmato i vecchi Apple II, qualche considerazione sui linguaggi disponibili, infine poche parole dedicate a chi fa sul serio.

13.1 Un nuovo Apple per vecchi applisti

Senza dubbio, una buona parte degli acquirenti di Apple IIgs è composta da appassionati della mela immortale che sono passati al nuovo computer abbandonando un vecchio modello. La maggior parte aveva imparato a programmare un poco, con le mani sulla tastiera ed un manuale Apple in grembo. Cosa si trovano davanti oggi queste persone?

Tutti i linguaggi di programmazione creati per i vecchi Apple II funzionano perfettamente con Phoenix: in più, la tripla velocità dà nuova vita ad idee messe nel cassetto come troppo impegnative.

E molto difficile, però, passare dai vecchi linguaggi – il Basic Applesoft, il Pascal UCSD – a quelli nuovi che permettono di struttare davvero Phoenix con la stessa facilità con cui siamo passati dal vecchio al nuovo modello.

Apple ha abbandonato da tempo il Basic Applesoft: la versione incorporata in Apple IIgs è fondamentalmente la stessa che venne creata nel 1980 per il modello Europlus, fatte salve un paio di modifiche striminzite, tipo la possibilità di battere i comandi in minuscolo.
E molto dubbio anche che venga ammodernato il Pascal UCSD: Apple ha abbandonato il linguaggio che solo lo scorso anno caldeggiava tanto per una serie di motivi teorici e pratici.

Oggi Apple, nei suoi manuali tecnici e nei suoi programmi preferisce usare il linguaggio C: la versione che Apple commercializza per Apple IIgs è particolarmente ben fatta e potente.
Scegliere di tagliare i ponti col passato piuttosto che voler rinverdire a tutti i costi due linguaggi creati tempo fa ha senso per due motivi: la ditta cui era stato commissionato il Pascal per Apple IIgs non ha portato a termine la commissione, ed Apple Computers ha dovuto distribuire, a fianco dell'Assembler, il solo linguaggio C. In questo senso la scelta è stata forzata.
In più dobbiamo ricordare che la scelta di compatibilità verso il basso ha sempre dei costi da pagare. Non c'è dubbio che Apple IIgs avrebbe potuto essere una macchina molto più potente e veloce se non si fosse ricercata la compatibilità con gli Apple IIc: di converso, si è scelto di ricominciare daccapo almeno col linguaggio di programmazione, per non pagare lo scotto del passato.
E una scelta analoga, da questo punto di vista, a quella compiuta nel 1983 abbandonando DOS per passare al ProDOS.

L'effetto pratico che questo ha sui vecchi appassionati di Applesoft e UCSD Pascal è chiaro: non sarà loro possibile usare a fondo la nuova macchina con i vecchi linguaggi.

Non sarebbe onesto chiudere ogni spiraglio. Come forse sapete, Applesoft Basic è stato creato ai suoi tempi da casa Microsoft: non escludo che Microsoft decida di produrre un nuovo Basic, analogo allo MS Basic per Macintosh, e che nel farlo scelga di mantenere stabili comandi e sintassi del vecchio Applesoft. Se io fossi lo sviluppatore di casa Microsoft, farei esattamente così. E corre voce che Microsoft abbia da mesi le mani in pasta nell'Apple IIgs, anche se non si è visto ancora nulla.
Per quanto riguarda il Pascal, le cose sono anche più semplici. Ad esempio, la TML, la ditta che ha prodotto il Mac Pascal per Macintosh e lo Instant Pascal per gli Apple II, ha commercializzato un ottimo Pascal che dispone praticamente di tutti i comandi già familiari sotto UCSD: ma attenzione, il sistema operativo è

il ProDOS 16, e il sistema di sviluppo è lo Apple Programmers' Workshop, quindi c'è da imparare ad usare un altro editore, un altro compilatore e un altro linker.

13.2 Nuovi linguaggi per nuovi e vecchi applisti

La produzione di linguaggi di programmazione nei primi mesi di disponibilità di Phoenix è stata entusiasmante, come ho già accennato altrove in questo libro.
Fatto sta che il microprocessore 65816 è semplicemente meraviglioso per quanto riguarda la creazione di linguaggi di programmazione, e questa bontà comincia a venire ripagata.

Il succitato Apple Programmers Workshop è un sistema di sviluppo completo che può venire usato con qualunque linguaggio di programmazione assemblato o compilato: chiunque voglia creare un linguaggio può fare in modo che esso sia aggiunto in modo semplice e nitido allo APW, e così un programmatore può far collezione di linguaggi soffrendo il minimo sforzo possibile per usarli tutti insieme, scegliendo il più adatto a seconda delle sue capacità e della necessità contingente (cioè a seconda del tipo di applicazione che vuole sviluppare).

In particolare, Apple ha distribuito un assemblatore 65816 che può anche essere usato per creare programmi per i più vecchi Apple, ed un compilatore C. Entrambi promettono bene. Tanto di cappello poi all'eccellente TML Pascal: questo linguaggio offre un compilatore velocissimo e tutto quanto serve per usare il toolbox. E anche possibile creare programmi in standard Pascal, senza tener conto delle caratteristiche di Apple IIgs: in quel caso il compilatore provvede ad usare il toolbox in modo perfettamente trasparente.
Tutti i tre linguaggi citati permettono di creare accessori di scrivania.
Ho visto annunci di compilatori Modula II, Fortran, Basic ed altro ancora, ma non mi sbilancio dato che non vi ho messo sopra le mani.
E molto importante in questo campo ricordare che con Apple IIgs è possibile creare un programma unendo routine create in linguaggi differenti: curare la parte matematica in Fortran, quel-

la logica in Pascal e i compiti che hanno bisogno di velocità in C o Assembler è semplice e relativamente veloce. I buoni programmatori potranno lavorare con molta efficienza quando avranno a disposizione tutti i compilatori necessari.

13.3 Per chi fa sul serio

Tra i miei lettori ci sarà senza dubbio qualche programmatore professionista, che pensa anche alla possibilità di commercializzare i programmi che crea.
Questo breve paragrafo è dedicato a lui.

Il mercato del software per Apple in Italia è stato sempe sottovalutato e maltrattato. Un po' per la presenza da sempre di una pirateria del software che, agguerrita e sofisticata, ha poco da invidiare a quella americana, un po' perché i distributori di software ormai non vedono altro che i PC IBM (i più lungimiranti danno qualche occhiata a Mac), chi voglia piazzare i propri prodotti non avrà la vita semplice.
A questo si aggiunge la discutibile politica di Apple Italia, che ha scelto di pubblicizzare (poco) l'Apple IIgs come una macchina da magazzino e ufficio con qualche applicazione ludica. la verità è che, da sempre, gli Apple II sono i computer più eclettici presenti sul mercato: buoni giochi, buone utilità, buoni linguaggi, e anche qualche buon gestionale – tutto buono o molto buono, nulla di eccellente ma anche niente talloni d'Achille, che tutti gli altri PC hanno.

Di conseguenza, mentre negli States gli Apple IIgs si vendono come il pane e i programmi possono vedere centinaia di migliaia di copie (Copy II Plus ha superato le duecentomila, AppleWorks viaggia sul doppio di quella cifra), in Italia è persino difficile trovare qualcuno che venda i programmi più importanti.
Invito, sia a titolo personale, sia a nome della rivista Bit, chi avesse qualche idea per migliorare la situazione a farsi vivo.

Per quanto riguarda la creazione del software, ci vuole del tempo per imparare ad usare al pieno delle sue possibilità il toolbox.Dato che l'enfasi è stata posta sulla facilità d'uso per l'utente, parte del peso in più cade sul programmatore che voglia realizzare programmi professionali. Si tratta di costruirsi con un po' di pazienza qualche libreria per l'uso più comune delle finestre, spazi di dialogo e cose simili, e poi si può procedere con buona velocità e risultati sorprendenti.

Nel creare programmi da commericalizzare, basta tener presente che da sempre Apple è un computer per tutti, mentre da oggi è anche un computer che tutti debbono saper usare senza leggere una riga di documentazione. La lettura del manuale intitolato Apple Human Interface Guidelines è consigliabile: tra molte cose risapute vi si trovano piccole perle di saggezza.

Citerò, da un documento Apple mai pubblicato, un caso significativo. I programmatori del dimostrativo intitolato "Apple presenta l'Apple IIe" avevano realizzato un ottimo prodotto. L'unico punto irrisolto era relativamente semplice: dato che le persone che avrebbero usato il programma (possibili acquirenti nel negozio, neo proprietari a casa, e intere scolaresche durante le visite guidate) avrebbero potuto indifferentemente usare un monitor a colori o in bianco e nero (o verde e nero, o giallo e nero, eccetera) era necessario che il programma chiedesse quale delle due possibilità era vera e si regolasse di conseguenza.

Come primo tentativo, si scelse di disegnare un grafico a colori e di chiedere all'utente se avesse un monitor a colori. Il risultato fu miserevole, perché molti utenti pensavano che forse il loro monitor era a colori ma con il contrasto abbassato; come potevano sapere che la cosa era irrilevante?

Come secondo tentativo, sullo schermo vennero stampate le parole ROSSO VERDE GIALLO BLU nei rispettivi colori, e la domanda divenne "Queste parole sono colorate?" Com'era intuibile, tutti quelli che stavano usando un monitor verde e nero o giallo e nero risposero di si.

Si scelse allora un diversivo: la domanda divenne "Queste parole sono disegnate in più di un colore?" E tuttavia ci fu ancora qualche errore, perché qualcuno, con un monitor verde e nero, fece notare che sul suo monitor c'erano due colori, il verde ed il nero.

Anche se gli autori del programma erano a questo punto del parere che la soluzione migliore sarebbe stata di regalare a tutti gli interessati un monitor a colori, si provò con la scritta "Le parole che vedete sono in molti colori differenti?" 'Disgraziatamente, parecchi utenti con monitor monocromatico risposero affermaticamente, perché in quella lunga domanda credettero di leggere "Le parole che vedete sono molti colori differenti?".
Infine il problema venne risolto chiedendo se "Queste parole appaiono in molti colori?".

In conclusione, fare un programma semplice da usare può essere un bel lavoro.

14

DI QUI ALL'ETERNITÀ

Dopo tutto quello che abbiamo visto, ci resta solo un argomento da toccare: le prospettive di crescita dell'Apple IIgs. E non voglio parlare di memoria o di periferiche, perché ne abbiamo già accennato e perché si tratta di discorsi piuttosto risaputi: la domanda è invece "quali e quante migliorie restano da fare, e possiamo aspettarci verranno predisposte, nel prossimo futuro".

14.1 Il coprocessore matematico

Abbiamo visto che il cuore e cervello dell'Apple IIgs è il suo microprocessore, chiamato 65816. La moderna tecnologia dei processori ci mette a disposizione anche degli aiuti ai microprocessori, i cosiddetti **coprocessori**.

Un coprocessore è a pieno diritto un microprocessore a sua volta, che viene fatto agire ad una velocità molto maggiore di quella del processore principale e che usa parole di una lunghezza maggiore: come dire che, mentre il 65816 manipola quantità di sedici bit alla volta, un coprocessore accoppiato potrebbe surclassarlo usando sessantaquattro bit la volta.

Un coprocessore non è eclettico come un normale processore: tutte le operazioni che è in grado di svolgere sono operazioni matematiche. Somme, sottrazioni, moltiplicazioni e divisioni

vengono risolte al ritmo di milioni al secondo; le operazioni più complesse vengono eseguite in tempi maggiori, ma pur sempre di qualche ordine di grandezza migliori dei tempi richiesti quando i calcoli vengono eseguiti sul 6516, che è in grado di eseguire direttamente solo somme e sottrazioni, e deve ricorrere ai sottoprogrammi della libreria SANE per eseguire i calcoli più complessi.

Vediamo come in tre differenti casi – computer con solo processore, computer con SANE, computer con coprocessore – vengono risolti i problemi matematici.

Normalmente, su un computer qualsiasi (diciamo per esempio su un Apple IIe) ogni qual volta un programma deve risolvere qualche operazione matematica, il programmatore deve scrivere una routine matematica in grado di far fronte al problema.

Su Apple IIgs un programma in ROM, il SANE (ne abbiamo parlato nel capitolo 9) risolve questo tipo di problemi per il programmatore, rendendo disponibile un modo flessibile e completo di trattare i calcoli. È un passo avanti.

D'altra parte, un computer dotato di coprocessore (per esempio un PC IBM per il quale venga acquistato, separatamente, il coprocessore 8087), può metterlo a disposizione del programma che sia in grado di accertarne la presenza, e sviluppare una potenza molto superiore i quella di ogni altro computer (Phoenix compreso). I migliori programmi del PC IBM svolgono l'ingrato compito di cercare il coprocessore, anche se debbono sempre e comunque contenere sottoprogrammi per sbrigarsela da sé se non lo trovano... La maggior parte dei programmi, data l'intrinseca difficoltà di tenere il piede in due scarpe, rinuncia alla possibilità e lascia l'eventuale coprocessore inutilizzato.

Possiamo immaginare un quarto caso. Immaginiamo che un coprocessore divenga disponibile per Apple IIgs e che voi ne decidiate l'acquisto.

Dato che i creatori di Phoenix conoscono il loro mestiere, hanno creato SANE avendo già in mente questa possibilità: SANE può riconoscere autonomamente la presenza di un coprocessore, e dirottare verso di questo tutti i calcoli veri e propri, senza che il programma che ha chiamato SANE debba accorgersene.

Questo significa che tutti i programmi, indifferentemente, vecchi e nuovi, si ritroveranno ad essere più potenti e precisi e veloci nei calcoli matematici, ed i programmatori non dovranno

mai preoccuparsi di gestire personalmente le chiamate al coprocessore.

Attendete il coprocessore prossimamente su questi schermi...

14.2 Il clock più veloce

Visto che abbiamo pensato ad aiutare il 65816, non potremmo pensare di renderlo più veloce?

Ogni microprocessore è caratterizzato dalla cosiddetta **velocità di clock**, la velocità che scandisce i tempi del microprocessore e stabilisce ogni quanto tempo viene eseguita una istruzione.

Nell'Apple IIgs il clock può essere disposto a due differenti velocità dal Pannello di Controllo. La velocità Normale corrisponde a 1 MhZ, quella Veloce a 2.8 MhZ.

MhZ è il simbolo di Megahertz: un megahertz corrisponde ad un milione di cicli al secondo. Tanto per saperlo, il PC IBM viaggia con un clock di 4.77 MhZ, lo Apple Macintosh a 7.5 MhZ: ma non prendete queste misure come indicative della velocità del computer, perché questa è fatta anche della velocità dei dischi (i dischi del IIgs sono molto più veloci sia di quelli di IBM sia dei loro stessi fratelli in versione Mac), della capacità dei chip schiavi, dal numero di differenti modi di indirizzamento (ben ventiquattro per 65818) della potenza delle singole istruzioni del processore (per esempio lo 8086 del PC IBM è decisamente negato per l'uso della grafica), eccetera.

In teoria, comunque il 65816 potrebbe essere spinto sino alla velocità da favola di 8 MhZ. Perché non è stato fatto da subito?

Il problema non è solo di processore. Il problema maggiore sta nella RAM, la memoria di lettura e scrittura di cui il computer è dotato.

La gran parte delle operazioni del microprocessore richiedono un accesso in memoria: la memoria deve essere abbastanza veloce da poter tener dietro al 65816, oppure tutta la velocità di questo sarebbe perfettamente inutile. Immaginatevi di correre ad imbucare una lettera: che senso ha se il postino non passerà prima di mezza giornata? Alla stessa stregua, è inutile rendere più veloce il microprocessore se poi esso dovrà passare tutto il suo tempo ad aspettare che la RAM gli dia la risposta che sta aspettando.

Il prezzo della memoria RAM dipende da due fattori: la dimensione (perché, ovviamente, un chip da 256 Kbyte costa più di un chip da 64 Kbyte – anzi, un chip da 256 Kbyte costa di più di 4 chip da 64 Kbyte) e la velocità di risposta alle richieste.
Una memoria dieci volte più veloce di un'altra a rispondere può costare cento volte di più. A queste condizioni potete ben immaginare che la scelta dei progettisti dell'Apple IIgs è ben pensata.

Ma non disperiamo. Il prezzo dei chip più veloci è in continua discesa, e certo un bel giorno verrà in cui...

14.3 La multiprogrammazione

Un passo di notevole progresso, di indiscussa potenza, sarebbe la disponibilità della multiprogrammazione sul nostro computer.
Per **multiprogrammazione** (talvolta indicata anche con il termine inglese *multitasking*) intendiamo la possibilità di far eseguire dal computer più programmi simultaneamente; in realtà non è possibile che i programmi vengano veramente fatti procedere in parallelo, per il semplice motivo che il microprocessore può soltanto eseguire le istruzioni una alla volta. Tuttavia, è possibile dare l'illusione di una elaborazione parallela grazie alla stupefacente velocità di esecuzione del processore.

Normalmente il 65816 si dedica ad un solo programma, eseguendolo sequenzialmente, una istruzione alla volta, dall'inizio alla fine. È tuttavia possibile interrompere questa sequenza mandandogli un segnale elettrico (un **interrupt**) che, gestito sotto il controllo del sistema operativo, indica al 65816 di passare ad un altro punto e di procedere l'elaborazione da lì. Ad esempio, il mouse è una periferica *interrupt-driven*, cioè governata grazie agli interrupt: ogni qual volta viene spostato viene generato un interrupt al 65816 che è tenuto a servire il mouse prima di poter proseguire con quello che stava facendo (questo si rende necessario perché se non viene concessa l'attenzione del processore subito dopo un movimento non è possibile rilevare con esattezza l'ampiezza ed il verso del movimento. È per questo motivo che durante le fasi di uso del disco, quando gli interrupt non

possono venire serviti, il mouse diventa tanto legnoso ed impreciso da usare se non addirittura bloccato).

Se esiste entro il computer un generatore periodico di interrupt (un orologio), sotto il controllo del sistema operativo la generazione (ogni mezzo secondo o giù di lì) di un interrupt può segnare il tempo per il processore, che concederà mezzo secondo del suo tempo ad un programma, poi un altro mezzo secondo ad un altro programma, poi un altro mezzo secondo ad un terzo programma, ed infine ricomincerà la sequenza dal primo programma. La sequenza si ripete sinché uno dei programmi non termina (e dunque non richiede più l'uso del 65816) oppure sinché un nuovo programma non viene inserito nella lista dei programmi in esecuzione (nel qual caso tutti gli altri tre riceveranno un poco meno spesso i servigi del microprocessore e saranno quattro programmi ad avanzare simultaneamente).

L'unico limite al numero di programmi che possono essere usati contemporaneamente è dovuto alla quantità di memoria disponibile.

Si tratta di una spiegazione approssimativa di un principio piuttosto complesso e di grande potenza. Tutti i lettori a cui l'argomento interessa possono rivolgersi al libro **Conoscere i sistemi operativi**, che spiega nei dettagli il funzionamento dei sistemi operativi.

A cosa serve la multiprogrammazione? In primo luogo essa è utile quando si vuole far stampare un documento dalla stampante: normalmente il computer resta bloccato con il microprocessore assorbito a servire la lenta stampante. Con la multiprogrammazione, solo una piccola parte del tempo del computer viene dedicata alla stampante, ed è possibile procedere ad altri compiti durante l'esecuzione della stampa.

Meglio ancora, esistono numerosi compiti che possono venire eseguiti in *background*, cioè in sottofondo, mentre voi state usando normalmente il computer per lavorare: per esempio, la compilazione di un programma, oppure la riordinazione dei dati di un data base o magari il controllo dello status di un disco sospettato di avere un guasto... e chi più ne ha più ne metta.

Cosa è indispensabile per poter avere la multiprogrammazione su un computer?

In primo luogo, un microprocessore abilitato a ricevere gli interrupt (e il 65816 può farlo tranquillamente).

Poi, è necessario un ente che scandisca gli intervalli di tempo

per il microprocessore: Apple IIgs dispone di un orologio incorporato, e così anche questa necessità è soddisfatta.

La terza necessità è più intangibile. I programmi, tutti i programmi debbono essere rilocabili: questo significa che debbono poter essere eseguiti in un qualunque punto della memoria RAM del computer. Questa richiesta non è vitale in un computer monoprogrammato, perché un programma può richiedere per sé qualunque punto della memoria; nel caso della multiprogrammazione invece il programma deve poter funzionare ovunque, per evitare la collisione di aree di memoria usate da programmi coesistenti in memoria, che impedirebbe ad entrambi di poter funzionare. Si dà il caso che, grazie al potente Memory Manager dell'Apple IIgs ed alla saggia scelta effettuata nel creare ProDOS 16, i programmi creati specificamente per Apple IIgs siano tutti rilocabili.

Esiste una quarta condizione non vitale: ogni programma dovrebbe essere "rientrante" per facilitare le cose – non ne parliamo qui perché si tratta di un argomento spinoso.

Infine, è necessario che un sistema operativo molto sofisticato e creato esplicitamente per mettere a disposizione la multiprogrammazione sia in controllo del sistema fisico.

Al momento, il sistema operativo dell'Apple IIgs, il ProDOS 16, non è in grado di gestire la multiprogrammazione (anche se ci sono cenni incoraggianti in quel senso: avrete certo notato che con Phoenix è possibile proseguire nell'uso della macchina anche quando la stampante sta funzionando, un inizio di multiprogrammazione). Quando però, tra non molto tempo, la dotazione standard di memoria RAM sarà più alta dei 256 Kbyte che oggi costituiscono il minimo degli Apple IIgs, ci sarà più spazio per il sistema operativo, che potrà divenire più complesso e quindi metterci a disposizione la multiprogrammazione. Si tratta, in un certo senso, della miglioria più facile ad ottenersi tra quelle di cui abbiamo parlato sin qui, e non dubito che da qualche parte, qualcuno tra i programmatori della Apple Computer ci stia pensando in questo stesso momento.

14.4 Il direct memory access

Uno stratagemma che rende molto più veloce un computer è

quello che va sotto il nome di **direct memory access**, che si traduce in "accesso diretto alla memoria". Di cosa si tratta?

Quando andiamo ad analizzare come il microprocessore 65816 passa il suo tempo, notiamo immediatamente due attività che lo costringono ad attendere senza poter fare nulla.

In realtà, il "cervello" di Apple IIgs passa la grande maggioranza del suo tempo senza fare nulla per un semplice motivo: sta attendendo che noi lentissimi esseri umani ci decidiamo ad impartirgli un ordine. In questo momento, per esempio, io sto battendo sulla tastiera del mio Apple IIgs le lettere che voi leggerete. Dato che sono un dattilografo piuttosto veloce posso battere, diciamo, cinquanta parole al minuto, ovvero pigio un tasto ogni quinto di secondo.

Però, il 65816 è in grado di compiere 2.800.000 microoperazioni ogni secondo, e dunque si ritrova con un dato da elaborare (un tasto che ho premuto) ogni centomila tentativi circa. Non male, no?

D'altra parte, non posso proprio scrivere centomila volte più veloce di quanto sto facendo ora... è il computer che deve servire me, e non viceversa, e quindi che sia costretto ad aspettarmi non mi preoccupa. Quello che è noioso è il momento in cui sono io a dover aspettare lui.

Il secondo maggior motivo di perdita di tempo può essere rimediato. Si tratta degli accessi al disco: il disco non è centomila volte più lento del computer, come lo sono io, ma è pur sempre di qualche migliaio di volte troppo lento. Ogni qual volta Apple IIgs richiede dei dati al disco deve aspettare un tempo (per lui) lunghissimo prima di avere la risposta.

Con il direct memory access, le cose cambiano: il microprocessore del computer non dovrebbe più stare ad aspettare che il disco gli risponda, ma si limiterebbe a spiegargli cosa vuole che sia letto e dove – nella memoria RAM – vuole ritrovarselo. Poi procede a fare qualche altra cosa, e così non perde tempo e non ne fa perdere a me. Quando il disco ha terminato le operazioni manda un interrupt al processore per segnalare che ha finito e che i dati sono disponibili.

Perché si possa avere direct memory access servono dei disk drive molto sofisticati, che per il momento hanno costi troppo elevati. D'altra parte, in futuro anche questi saranno disponibili: ma potranno venire usati sul nostro Apple IIgs?

La risposta è un gioioso SI. Perché un disk drive con direct me-

mory access possa venire collegato ad un computer, deve essere disponibile un punto di ingresso alla memoria del computer (quello che in gergo si chiama il **bus della memoria** del computer). Ora, si dà il caso che attraverso uno qualunque degli slot di Apple IIgs sia disponibile il bus del sistema, e dunque tutto quello che dobbiamo aspettare è che qualcuno trovi il sistema per rendere quei disk drive alla portata delle nostre tasche e un poco ancora perché Apple Computer adatti il ProDOS ad usare i nuovi dischi. Il secondo passaggio richiederà pochissimo tempo, perché SmartPort (ne abbiamo parlato altrove nel libro) è già da ora in grado di gestire i dischi in DMA ad un eccellente livello di sofisticazione.

Succederà, succederà...

14.5 Il processore 65832

Ho tenuto il jolly per ultimo: si tratta di una nuova rivoluzione che renderebbe un ulteriore modello di Apple II tanto più potente dell'Apple IIgs quanto Phoenix è meglio dei modelli precedenti.

Abbiamo più volte accennato in questo libro che il processore 65816 è basato sul precedente 6502, un processore più semplice ad otto bit, nel senso che è in grado di eseguire tutte le operazioni che il 6502 sa fare e ne può eseguire altre in più. Il 6502 stava al cuore dei precedenti modelli di Apple II ed è soprattutto grazie a questa **compatibilità verso il basso** che il 65816 permette ad Apple IIgs di poter eseguire tutti i programmi creati per i suoi fratelli precedenti.

Ora, fatto sta che la corsa alla miglioria non è per niente terminata. Il signor William D. Mensch, uno degli ingegneri progettisti che crearono il processore 6502 presso la compagnia MOS technology, è passato in seguito a fondare la compagnia The Western Design Center Inc. e qui ha iniziato gli studi che hanno portato alla creazione del nostro formidabile 65816.

In questo periodo l'infaticabile ingegner Mensch sta pensando a battere nuovamente sé stesso, e cerca di creare un nuovo processore che sarà conosciuto con la sigla 65832. Le prime due cifre stanno ad indicare la famiglia a cui appartiene: questo nuo-

vo microprocessore dovrebbe saper fare tutto quello che 65816 sa fare.

Ma il nuovo processore sarà un trentadue bit, come il favoleggiato Motorola 68000 che dà vita al Macintosh, e a sua volta introdurrà nuove e più potenti possibilità per i programmatori, compiendo così un ulteriore balzo in avanti. Inoltre, il 65832 incorporerà un coprocessore matematico.

Quando e se tutto ciò diverrà realtà non posso davvero dirlo. Il passato consiglia la prudenza: il 65816 è stato annunciato per disponibile sin dal giugno 1984, ma una serie di innumerevoli ritardi ha fatto sì che solo ai primi dell'87 abbiamo potuto mettere le mani sul nostro insostituibile Phoenix.

Non posso neppure dire se la sostituzione sarà indolore.

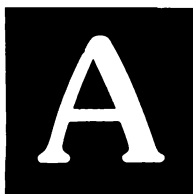
Il dilemma sta nella piedinatura del microprocessore, cioè nella disposizione di quelle presine metalliche che ne escono e che permettono di collegarlo al mondo esterno: se il 65832 potrà essere creato **pin-compatibile**, cioè con lo stesso numero di piedini, disposti nello stesso ordine e con lo stesso significato, rispetto al 65816, allora sarà possibile sostituire il microprocessore dentro ad Apple IIgs con spesa minima.

Ma se non sarà possibile, allora non dubito che Apple produrrà un ulteriore modello della serie II, e ci toccherà di nuovo cambiare una macchina per una nuova.

Nessuna delle due possibilità va esclusa. Quando uscì il 65C02, una miglioria del 6502 che è usata da Apple IIc, la sua pin-compatibilità col vecchio modello rese possibile la sostituzione del processore dentro ai vecchi Apple IIe. Al contrario, 65816 non è pin-compatibile con 65C02, e dunque gli Apple IIc ed Apple IIe non possono venire trasformati in Apple IIgs con quella stessa facilità.

In conclusione, ci vorrà un po' prima che l'Apple II possa trasformarsi in un computer a trentadue bit, ma non dubito che prima o poi accadrà.

Perché, come grida a pieni polmoni la pubblicità americana del nostro sfavillante Phoenix...

Apple II forever!, per sempre. E, dunque, non addio ma arrisentirci.

APPENDICE

Il primo Apple II non portava lettere che lo distinguessero: era **lo** Apple II. Tuttavia, una serie di piccole modifiche hanno fatto sì che venisse distinto in otto sottospeci (le revisioni). Il primo Apple II (il glorioso **revision zero**) veniva venduto con 4 Kbyte di RAM, 8K di ROM contenenti il Vecchio Monitor (compresi Sweet Sixteen, Programmer's Aids e Mini Assembler, e con le istruzioni Snglestep e Trace) ed Integer Basic. Aveva otto slot di espansione.

All'accensione, il computer era in Monitor. Era necessario eseguire una serie di operazioni per entrare in Basic, ed il RESET portava sempre in Monitor (non era possibile programmarlo).

Apple II aveva solo il set di lettere maiuscole, non le minuscole. L'alta risoluzione (sempre che aveste aggiunto i chip di memoria RAM necessari) era rigorosamente in bianco e nero: fu con il **revision one** che, grazie ad un trucco, si ottennero i sei colori – ecco perché è sempre stato più difficile usare l'alta risoluzione a colori sugli Apple II che sui computer di concezione più recente.

Il modello successivo fu lo **Apple II**+ (Plus). La ROM era salita a 12 Kbyte, e conteneva il nuovo Applesoft Basic di Microsoft, e il Monitor Autostart (il che significa che la macchina era in Basic all'accensione, e che lanciava automaticamente il disco se presente). La RAM era di 48 Kbyte. L'Apple II+ andava anche conosciuto come **revision seven** e fu il primo Apple II ad essere importato in Italia – dalla Iret Informatica, che solo più tardi fu acquistata dalla Apple americana e divenne Apple Computer Italia SpA.

Con il primo **Apple IIe** (Enhanced), Apple introdusse le minuscole di standard nel set di caratteri, e modificò la tastiera per farla divenire ASCII-completa (un termine tecnico che significa "in grado di generare l'intero set di caratteri ASCII"). La ROM salì a 16 Kbyte: la novità rilevante stava in un programmino di autodiagnosi. La RAM era di 64 Kbyte, comprendendo così in dotazione standard la cosiddetta language card, la zona di memoria che condivide gli indirizzi del Basic – e che dunque Basic non usa, ma viene usata da Pascal e ProDOS. Spariva così lo slot di espansione numero zero, riservato alla language card, ma appariva il cosiddetto slot ausiliario destinato alla scheda ottanta colonne.

Apparvero i tasti mela-vuota e mela-piena, collegati ai pulsanti zero ed uno del joystick. Con Control-Mela Vuota-RESET un pro-

Fig. 3. Apple II e.

grammino in ROM fa credere al computer di essere stato appena acceso, mentre Control-Mela Piena-RESET invoca il test di autodiagnosi.

I circuiti integrati che affollavano a centinaia la scheda madre del II+ lasciavano il posto ad un numero limitato di circuiti custom (cioè "fatti apposta").

Apple IIe ebbe due revisioni, chiamate A e B. Nella revisione B venne introdotta la doppia alta risoluzione di 580x192 pixel in 16 colori. La terza revisione trasformò completamente il IIe, e ne parleremo tra poco.

Con **Apple IIc** (Compact) Apple Computers decide di far diventare standard tutte le schede di espansione più diffuse, memoria compresa (la RAM passa a 128 Kbyte,, il massimo che il processore possa "vedere" – in due blocchi però – senza dover ricorrere a trucchi di scomodità inaudita).

Per la prima volta, Apple II va incontro ad una revisione totale, un ripensamento che tocca ogni suo aspetto: è solo grazie al genio riconosciuto di Steve Wozniak che il nuovo computer risulta compatibile con il vecchio.

Cambia il processore – con un modello leggermente più po-

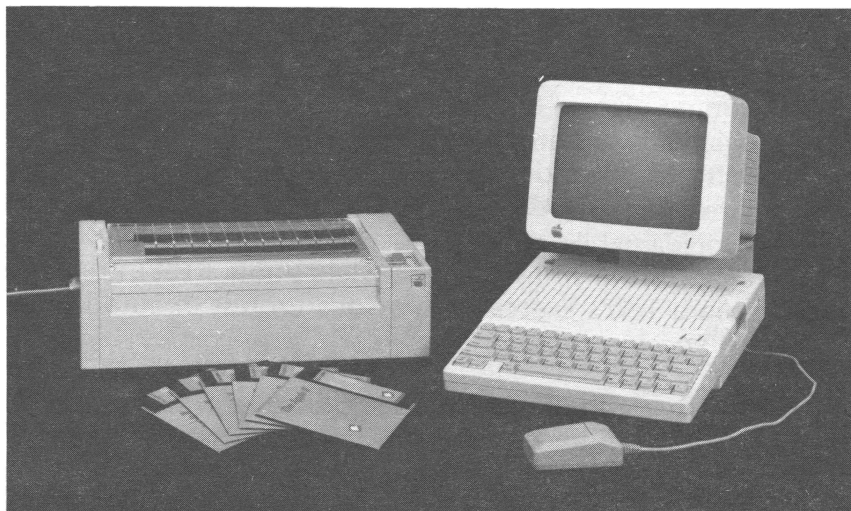



Fig. 4. Apple II c.

tente e che consuma pochissima energia – cambiano i circuiti custom che diventano solo cinque (IWM Integrated Wozniak Machine, IOU Input Output Unit, MMU Memory Management Unit, TMG Timing Generator, GLU General Logic Unit), la ROM supporta pienamente gli interrupt al processore (e il mouse diventa di casa), il disco è incorporato e sparisce l'uscita per le cassette, il display a ottanta colonne è pienamente gestito (le routine del IIe hanno un paio di banchi colossali e sono comunque insufficienti). Si aggiunge la possibilità del display di icone nel modo testo: i caratteri del mouse o MouseText Characters.

Premendo Control-Mela Piena-Mela Vuota-RESET si attiva Teri's Exercise, un programmino assolutamente inutile che ha il solo scopo di riempire qualche byte sprecato di ROM.

A questo punto Apple IIe è sopraffatto, ma gli viene concessa una ripulita perché possa restare in vita: è la **revisione C**, disponibile anche per i vecchi modelli acquistando lo enhancement kit. Il nuovo modello, con ROM molto simili a quelle del IIc e il processore 65C02, viene soprannominato **Eve: IIe Very Enhanced**. Viene venduto con la scheda 80 colonne, mouse e disco fornite: può usare i MouseText e gli interrupt: in cambio sparisce il programma di diagnosi.

Pochissimo dopo, anche il IIc va incontro a una revisione (anche per lui la revisione C, per una coincidenza), che lo lascia in grado di usare gli Unidisk da 800K e la rete AppleTalk. La revisione, in questo caso, tocca solo le ROM, che sono raddoppiate in capienza: la nuova ROM viene fornita a prezzo di costo anche per i vecchi modelli (la chiamano **Kit di espansione IIc** in Italia). Il nuovo IIc ottiene il *nickname*, o soprannome, di **Jane**, ma non chiedetemi il perché.

Con sommo piacere di programmatori e hackers, nel Monitor (che si chiama Monitor 256K) del nuovo IIe del nuovo IIc sono reintrodotte le operazioni di Singlestep e Trace per seguire i programmi in linguaggio macchina. Riappare anche il Mini Assembler, richiamabile da Monitor con il comando **!**.

Il comando Control-Mela Piena-Mela Vuota-RESET non corrisponde più a Teri's Exercise ma invoca un test di autodiagnosi (come sul vecchio IIe).

L'ultimo modello della serie è lo **Apple IIgs** (Graphic & Sound), ed è una nuova rivoluzione copernicana. Delle capacità del IIgs abbiamo parlato in tutto il resto del libro.

I cinque chip del IIc sono diventati uno solo: **Mega II** com-

prende in se tutte le funzionalità che nel vecchio Apple II+ richiedevano più di cento circuiti integrati.

Il Monitor è ancora espanso, ma sono tolte le opzioni di singlestep e trace.

La ROM è di 128 Kbyte, la RAM parte da 256 K e va sino a 4096 Kbyte (4 Mbyte). In futuro potrà arrivare a 8 e poi 16 Mbyte.

Ci sono di nuovo otto slot, con lo zero ancora una volta disponibile alle espansioni di memoria.

Appare Control-Mela Vuota-ESC, che chiama il pannello di controllo.

Nelle tabelle riassuntive trovate la storia, ben lungi dalla fine, di quel rissoso, irascibile, carissimo Apple II.

Notate che è possibile distnguere tipo di System Monitor con un PEEK alla locazione F8VERSION, indirizzo $FBB3, ed il modello di Apple II osservando IDBYTE a $FBC0 e SUBID a $FBBF.

E possibile dostinguere i modelli di Apple II ad 8 bit di processore da quelli a 16 bit (per il momento il solo IIgs) chiamando la routine $FE1F. All'ingresso il flag di carry deve essere settato, all'uscita è settato solo se ci si trova su un Apple II ad 8 bit. In caso contrario, vengono comunicate le caratteristiche del sistema come da tabella 3.

Modello	**Apple II**	**Apple II+**	**Apple IIe**	**Apple IIc**
ROM minima	8 Kbyte	12 Kbyte	16 Kbyte	16 Kbyte
ROM massima	14 Kbyte	14 Kbyte	16 Kbyte	32 Kbyte
RAM minima	4 Kbyte	48 Kbyte	64 Kbyte	128 Kbyte
RAM massima	64 Kbyte	64 Kbyte	128 Kbyte	128 Kbyte
Processore	6502	6502	6502A	65C02
Slot 0			Language Card	Language
Card				
Slot 1				Seriale RS232C
Slot 2				Seriale RS232C
Slot 3			Ausiliario	80 colonne
Slot 4				Mouse
Slot 5				---nullo---
Slot 6				Disk 5" 1/4
Slot 7				---nullo---
Scritta startup	APPLE] [	APPLE] [	Apple] [	Apple //c
System Monitor	Vecchio	Autostart	Autostart	65C02
Byte $FBB3	$38	$EA	$06	$06
Byte $FBC0	$60	$EA	$EA	$00
Byte $FBBF	$2F	$EA	$C1	$FF
Display	Maiuscole	Maiuscole	+ minuscole	+ MouseText

Modello	IIe (Eve)	IIc (Jane)	IIgs (Phoenix)
ROM minima	32 Kbyte	32 Kbyte	128 K
ROM massima	32 Kbyte	32 Kbyte	1024 K
Ram minima	--come il primo Nc--		256 K
RAM massima			8 M
Processore	65C02	65C02	65SC816
Slot 0	L. Card	L. Card	
Slot 1		RS232C	RS422
Slot 2		RS232C	RS422
Slot 3 .	80 colonne	80 colonne	80 colonne
Slot 4	Mouse	Mouse	Mouse
Slot 5		SmartPort	SmartPort
Slot 6	Disk II	Disk 5" 1/4	Disk 5" 1/4
Slot 7		AppleTalk	AppleTalk
Scritta start	Apple //e	Apple //c	Apple IIgs
Sys. Monitor	Monitor 256K	Monitor 256K	Monitor 16
Byte $FBB3	$06	$06	$06
Byte $FBC0	$E0	$00	$EO
Byte $FBBF	$00	$00	$00
Display	---come il primo IIc---		+ 16 colori in modo testo

Tabella 2: i modi di display

Pixel		Nome	Dal	Colori	
40	24	Testo	II	–	(1)
40	48	Bassa risoluzione	II	16	
80	24	Testo ottanta colonne	IIe	–	
80	48	Doppia bassa risoluzione	IIe	16	
280	192	Alta risoluzione	II	6	
560	192	Doppia alta risoluzione	IIe	16	
320	200	Super alta risoluzione	IIgs	256	(2)
640	200	Doppia super alta risoluzione	IIgs	256	(3)

(1) Dal IIc comprende le icone MouseText. Dal IIgs, in 16 colori.
(2) 16 colori per riga, 256 sullo schermo, da una palette di 4096.
(3) 4 colori per riga (16 usando il dithering), 256 sullo schermo, da una palette di 4096.

Tabella 3: i risultati della chiamata a $FE1F

Accumulatore:	bit 6	Il sistema ha slot di espansione memoria?
	bit 5	Il sistema usa Integrated Wozniak Machine?
	bit 4	Il sistema ha un orologio?

	bit 3	Il sistema usa Apple DeskTop Bus?
	bit 2	Il sistema usa le RS422 Zilog SCC?
	bit 1	Il sistema ha slot di espansione?
	bit 0	Il sistema ha i port (interfacce incorporate)?
Registro Y	byte alto Machine ID: zero su Apple IIgs, sarà incrementato sulle future macchine della serie	
Registro X	byte basso Versione della ROM (zero sui primi Apple IIgs)	

BIBLIOGRAFIA

Purtroppo per quanti non conoscono l'inglese, la maggior parte della documentazione sui computer Apple II – e specialmente quella tecnica più sofisticata – non è disponibile in italiano.

Gran parte degli esaurienti manuali prodotti dalla Apple Computers vengono tradotti in lingua italiana, ma non tutti. E, comunque, i testi più significativi nella letteratura sui computer Apple non sono quelli della casa madre.

Con Apple IIgs vengono forniti alcuni manuali introduttivi: sono assolutamente banali e trascurabili. Esistono altri manuali Apple più approfonditi, ma al momento in cui scrivo non so ancora quali e quanti verranno tradotti.

Similmente, non esiste ancora letteratura tecnica prodotta da terze parti sull'Apple IIgs.

I nomi ed alcune brevi descrizioni dei manuali tecnici Apple si trovano a pagina 164 del Manuale Utente.

Sono particolarmente interessanti:

Apple IIgs Firmware Reference con la documentazione dei programmi in ROM
ProDOS 16 Technical Reference con la descrizione di ProDOS 16
Apple IIgs Toolbox Reference in due volumi.

Tutti i manuali Apple originali sono editi da The Addison Wesley Publishing Company.

Per quanti vogliono conoscere meglio il sistema operativo dell'Apple II ed il modo in cui funzionano i disk drive, il riferimento d'obbligo è a due libri che sono unanimamente considerati testi sacri da tutti gli esperti:

Don Worth e Pieter Lechner
Beneath Apple DOS
Quality Software

Don Worth e Pieter Lechner
Beneath Apple ProDOS
Quality Software
Reston Publishing Company, Reston, Virginia
A Prentice-Hall Company

Quanti desiderano imparare a programmare in linguaggio macchina possono trovare utilissimo il libro

Don Lancaster
Assembly Cookbook for the Apple II
Howard W. SAMS & Co. Inc, Indianapolis, Indiana

In generale tutti i libri di Don Lancaster, un guru dell'Apple II, sono molto raccomandabili.
Per proseguire ad imparare la programmazione Basic dopo aver completato il corso dell'**Introduzione all'Applesoft Basic** dovrete procurarvi i manuali Apple

Manuale di riferimento del programmatore Applesoft Basic
(titolo originale: **Applesoft Basic Programmer's Reference Manual**)

Applesoft Basic programming with ProDOS

Il linguaggio più efficiente e meglio adatto all'Apple Iigs è il linguaggio C. Il manuale Apple intitolato **Apple programmer's Workshop: C language** è necessario per imparare ad usare le librerie di collegamento con il toolbox e le altre caratteristiche del C su Apple IIgs.

Se invece desiderate imparare la programmazione con Pascal, la cosa migliore che possiate fare è cambiare idea.

Luca Stefano Accomazzi

Il Manuale dell' Apple II GS

Scritta da uno dei maggiori esperti italiani, questa guida all'Apple IIGS è ricchissima di notizie, consigli d'uso e trucchi, ma il suo maggior pregio è di spiegare, in uno stile chiaro e discorsivo, come funziona il nuovo computer Apple. Entrando nei dettagli quando necessario, partendo dove finisce il "manuale utente", questo libro descrive la logica che governa Apple IIGS, ne descrive i componenti hardware e software, ne sottolinea i pregi.
Il nuovo System Monitor, il Toolbox, i Manager, la sofisticata e nuova SmartPort, l'emulazione dei precedenti modelli di Apple II, vengono spiegati in modo che anche i meno esperti possano imparare ad approfittare delle incredibili risorse di questo calcolatore: in due capitoli appositamente dedicati, poi, i programmatori trovano una vera messe di consigli, spiegazioni, notizie tecniche.

SOMMARIO

- Hardware
- Pannello di controllo
- Disco RAM
- Finder Mouse Desk 2.0
- GS Paint 1.0
- GS Write 1.0
- Assemblatore
- Compilatori C e Pascal
- Modi grafici
- Doppia super alta risoluzione
- Sintetizzatore digitale
- Compattazione e animazione
- Disk drive
- ProFile
- DOS 3.2 e 3.3
- Pascal 1.0, 1.1, 1.2 e 1.3
- ProDos 8 1.0 e 1.1
- ProDos
- System Monitor
- Mappa di memoria
- Toolbox
- SANE
- Coprocessore matematico
- Multiprogrammazione
- Direct memory access

ISBN 88-7056-792-3

GRUPPO EDITORIALE JACKSON

L. 28.000 Cod. CC576

576

Il Manuale dell' Apple II GS